

Manual de usuario del Generador Estocástico de Series Sintéticas GAMWGEN

Alessio Bocco (boccoalessio@gmail.com)
Daniel Bonhaure (danielbonhaure@gmail.com)
Guillermo Podestá (gpodesta@rsmas.miami.edu)

09 de October de 2020

Abstract

Este paquete contiene la implementación de un generador estocástico diario y multi-sitio de series meteorológicas sintéticas. La generación de series sintéticas es un insumo básico para el análisis probabilista de las sequías y sus impactos en el sector agrícola. El generador desarrollado es muy flexible y capaz de generar secuencias de valores diarios de precipitación y temperaturas máxima y mínima. A partir de estas últimas se pueden derivar otras variables como la radiación solar y la evapotranspiración.

El generador tiene dos variantes, una permite generar series para localidades puntuales, generador local, y otra que permite la generación en grillas regulares o en localidades arbitrarias, generador espacial. Esto la de la posibilidad al usuario de generar productos a medida para distintas aplicaciones. Además, el generador es capaz de utilizar variables auxiliares que producen simulaciones condicionadas para su uso en conjunto con modelos de cambio climático o de pronósticos estacionales para evaluar impactos en el largo o corto plazo, respectivamente.

Esta flexibilidad se sustenta en el uso modelos generalizados aditivos (GAM) que permiten modelar con mucha precisión el comportamiento de las variables meteorológicas y capturar las propiedades estadísticas de los datos observados. La modelación de las variables meteorológicas se divide en dos: por un lado, la ocurrencia y monto de precipitación y por otro, las temperaturas máxima y mínima. La ocurrencia de precipitación se modela a través de un modelo probit mientras que al monto se lo hace a través de una distribución aleatoria Gamma. Para la temperatura se utiliza un modelo autorregresivo condicionado por la ocurrencia de lluvia. A su vez, estos modelos pueden ser espacialmente correlacionados con campos aleatorios Gaussianos que contemplan la variabilidad espacial y temporal regional.

Además del generador, se incluyen diagnósticos estadísticos y gráficos para verificar la bondad de ajuste estadística de los GAM y validar que las series sintéticas sean consistentes con los registros históricos. Los diagnósticos son exhaustivos e incluyen todas las propiedades de las series que podrían afectar el desempeño de las mismas durante el análisis probabilista.

Contents

1	Introducción	2
2	Fundamento	4
3	Tipos de series sintéticas	5
4	Metodología	8
4.1	Introducción a los Modelos Aditivos Generalizados	8
4.1.1	Interpretabilidad vs Complejidad	8
4.2	Modelación	14
4.2.1	Clima local: Ocurrencia de lluvia	15
4.2.2	Clima local: Montos diarios de lluvia	16
4.2.3	Clima local: Temperatura	17
4.2.4	Tiempo local: Para una estación	18
4.2.5	Tiempo local: Para una grilla	20
5	Aplicación	21
5.1	Instalar paquetes necesarios	22
5.2	Creación de directorios	22
5.3	Generación de series sintéticas para una sola estación meteorológica	22
5.3.1	Crear archivos de entrada	23
5.3.2	Series sintéticas estacionarias	26
5.3.3	Series sintéticas pseudohistóricas	42
5.3.4	Series sintéticas correlacionadas espacialmente	59
5.4	Generación de series sintéticas para una grilla regular	80
	References	109

¹ 1 Introducción

² El análisis de series climáticas es siempre un desafío y más aún en regiones donde la disponibilidad de las mismas es escasa. Este déficit es un problema especialmente importante para

⁴ la caracterización del riesgo de desastres naturales. Este tipo de aplicaciones demandan
⁵ largas series climáticas para la realización de análisis cuantitativos del riesgo de desastres
⁶ que son claves para la estimación de su recurrencia y de las pérdidas asociadas. Las series
⁷ climáticas deben capturar la variabilidad natural del clima de la región de interés.

⁸ Al tratarse de un insumo necesario para el análisis probabilista de sequías el Sistema de
⁹ Información sobre Sequías para el sur de Sudamérica (SISSA) desarrolló un generador de
¹⁰ datos climáticos sintéticos. Los generadores estocásticos de clima producen series diarias de
¹¹ variables climáticas con propiedades estadísticas consistentes a las de los datos observados.
¹² Las principales variables de importancia son temperaturas máxima y mínima y precipitación
¹³ diaria. En muchas regiones estos datos se encuentran incompletos o son directamente inex-
¹⁴ istentes. Los registros suelen tener una duración insuficiente o sólo estar disponibles agre-
¹⁵ gados mensualmente. La cobertura espacial es otro de los problemas más comunes ya que,
¹⁶ en general, las redes de observación meteorológica son poco densas aún en zonas donde la
¹⁷ información meteorológica es de vital importancia.

¹⁸ La mayoría de los enfoques tradicionales para la generación de series estocásticas están lim-
¹⁹ itados por su capacidad de generar datos solamente en localidades para las que se cuenta
²⁰ con observaciones (por ejemplo, donde existen estaciones meteorológicas). Otra desventaja
²¹ de algunas de estas herramientas (sobre todo los generadores no paramétricos basados en
²² remuestreo de observaciones) es que solamente pueden producir valores dentro del rango
²³ observado en el registro histórico. En este proyecto, el generador desarrollado se basó en el
²⁴ modelo diseñado por Verdin *et al.* (2016). Este generador estocástico diario multivariado
²⁵ produce series sintéticas de precipitación y temperaturas máxima y mínima. El generador de
²⁶ Verdin *et al.* (2016) utiliza cuatro modelos estadísticos para modelar la ocurrencia y montos
²⁷ de precipitación y temperatura máxima y mínima basados en Modelos Lineales General-
²⁸ izados (GLMs). Estos modelos son paramétricos y utilizan regresiones lineales entre las
²⁹ variables para modelarlas. El proceso de ocurrencia de precipitación se modela como una re-
³⁰ gresión probit mientras que los montos se ajustan a una distribución aleatoria Gamma. Las

³¹ temperaturas máximas y mínimas son consideradas variables autorregresivas condicionadas
³² por la precipitación. Sin embargo, una de las limitaciones fundamentales del generador de
³³ Verdin *et al.* (2016) es que las temperaturas máxima y mínima se generan en forma indepen-
³⁴ diente para cada día, por lo cual la amplitud térmica diaria simulada a veces no es realista.
³⁵ Los diagnósticos realizados en base al generador de Verdin *et al.* (2016) demostraron que
³⁶ algunas propiedades de las series sintéticas producidas no reflejaban fielmente características
³⁷ importantes para el análisis de las sequías, por ejemplo, la persistencia de secuencias de días
³⁸ secos y lluviosos. Por este motivo, en este trabajo solo se mantuvo la estructura general del
³⁹ modelo de Verdin *et al.* (2016) y se modificaron todos los algoritmos (modelos estadísticos)
⁴⁰ para obtener series sintéticas más consistentes con los registros históricos.

⁴¹ 2 Fundamento

⁴² A partir de los datos climáticos históricos comienza el proceso de ajuste de un modelo
⁴³ estadístico que permita representar el comportamiento de cada una de las variables para
⁴⁴ cada localidad. El generador está dividido en cuatro modelos aditivos generalizados: dos para
⁴⁵ modelar la precipitación y dos para las temperaturas máxima y mínima, respectivamente.
⁴⁶ El concepto principal detrás de la generación de series sintéticas es que cada valor de una
⁴⁷ variable puede ser considerado como la suma de una componente climática (clima local) y
⁴⁸ otra componente meteorológica aleatoria o tiempo local (Kleiber *et al.*, 2013), es decir

$$X_{i,s} = \text{clima local} + \text{tiempo local}$$

⁴⁹ donde $X_{i,s}$ corresponde al valor de la variable X en el día i en la localidad s ; clima local
⁵⁰ corresponde a un valor medio de la variable para el día i en la localidad s y tiempo local
⁵¹ corresponde a un estado particular de la atmósfera en el día i en la localidad s . El componente
⁵² climático es especificado a través del ajuste de cada uno de los cuatro modelos aditivos

53 del generador. El componente meteorológico corresponde a los residuos de los modelos (la
54 diferencia entre los valores históricos y el componente climático estimado), es decir, a la
55 variabilidad no explicada por los mismos. El proceso de ajuste culmina con los parámetros
56 ajustados para los cuatro modelos mencionados. Posteriormente, se generan datos para los
57 años a simular que corresponden a los valores medios de la distribución para cada día del año
58 (clima local). Luego, se simulan una serie de valores aleatorios, a partir de los residuos de
59 cada modelo, que corresponden a la variabilidad propia de cada realización (tiempo local).

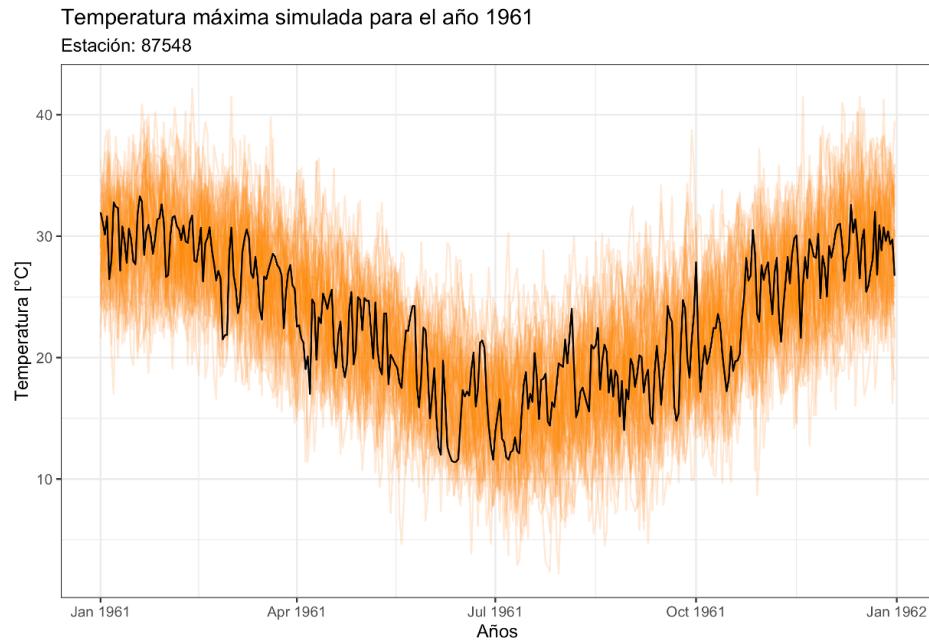


Figure 1: Componentes clima local y tiempo local

60 3 Tipos de series sintéticas

61 El generador GAMWGEN produce distintos tipos de series sintéticas. Desde el punto de
62 vista temporal, las series pueden ser **no condicionadas** (Figura 2.a), es decir, puramente
63 estacionarias; **pseudohistóricas** (Figura 2.b), que copian la variabilidad de baja frecuencia
64 y los cambios en la serie climática observada y **condicionadas** (Figura 2.c) que son series
65 forzadas a seguir la trayectoria de una salida de un modelo de cambio climático o una

66 trayectoria arbitraria definida por el usuario. Las siguientes Figuras muestran distintos
67 ejemplos de lo mencionado.

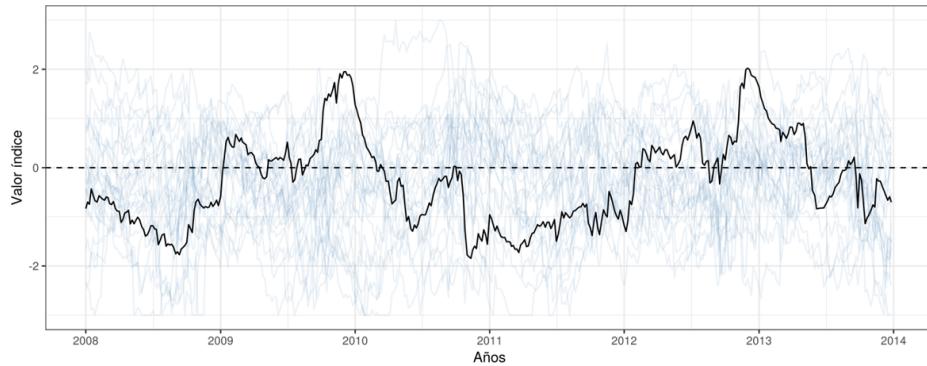


Figure 2: Tipos de series sintéticas: Series estacionarias

68 La línea negra corresponde a la serie temporal observada mientras que las distintas líneas
69 celestes corresponden a realizaciones del generador. Al tratarse de series puramente estocás-
70 ticas, son independientes entre sí.

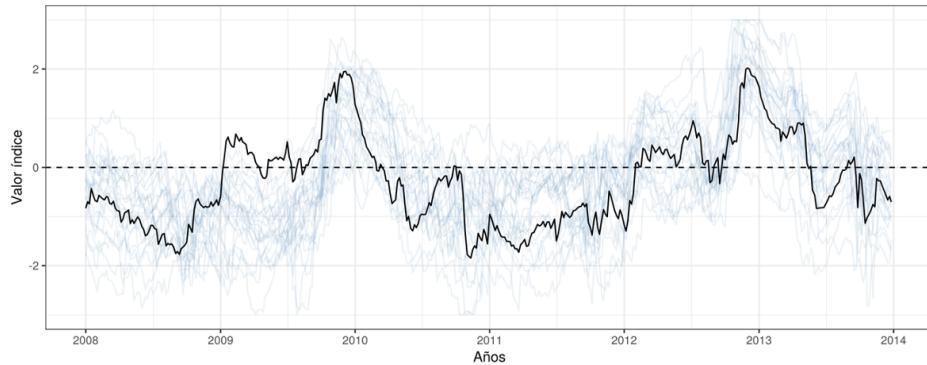


Figure 3: Tipos de series sintéticas: Series pseudohistóricas

71 La línea negra corresponde a la serie temporal observada mientras que las distintas líneas
72 celestes corresponden a realizaciones del generador. En este caso, el generador fue forzado
73 a seguir la trayectoria en los datos observados, por lo tanto, las series generadas siguen las
74 variaciones de los datos observados.

75 En esta configuración el generador sigue una trayectoria arbitraria elegida por el usuario.
76 En este caso es una trayectoria lineal por lo que la precipitación aumenta un determinado

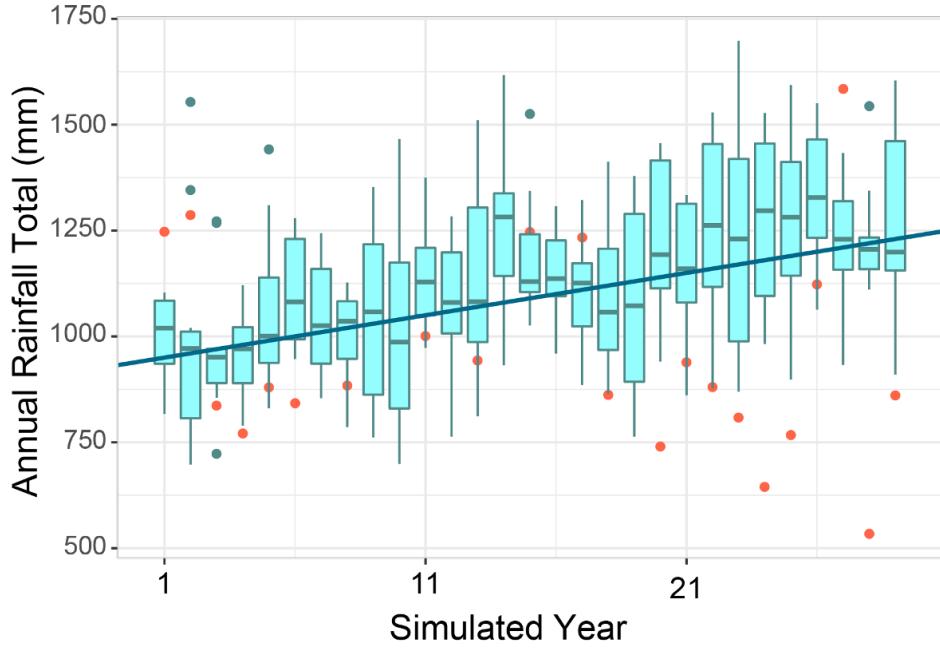


Figure 4: Tipos de series sintéticas: Series condicionadas

⁷⁷ porcentaje por año de manera lineal.

⁷⁸ Desde el punto de vista espacial se pueden generar series en estaciones meteorológicas, en
⁷⁹ puntos que no se correspondan con estaciones o en una grilla regular. La Figura 3 muestra
⁸⁰ un ejemplo de la generación en grillas sobre el Paraguay.

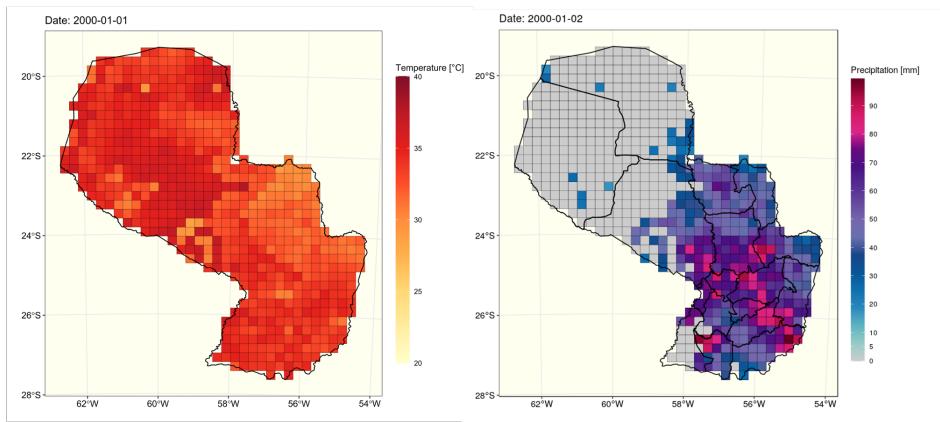


Figure 5: Tipos de series sintéticas: Datos grillados

⁸¹ El panel de la izquierda muestra la temperatura máxima del día 1 de enero de 2000 para todo
⁸² el territorio del Paraguay mientras que el mapa del panel derecho muestra la precipitación

⁸³ diaria para el mismo día.

⁸⁴ 4 Metodología

⁸⁵ Como su nombre indica, el generador está basado en Modelos Generalizados Aditivos (GAM,
⁸⁶ por sus siglas en inglés). Como se mencionó en la Introducción, este generador está basado
⁸⁷ en uno similar desarrollado por Verdin *et al.* (2016). Dicho generador estocástico utilizaba
⁸⁸ modelos lineales generalizados (GLM, por sus siglas en inglés). Los GLM son modelos muy
⁸⁹ interesantes ya que son fatalmente interpretables aunque carecen de la flexibilidad necesaria
⁹⁰ para capturar complejos patrones como las variaciones estacionales. Por ello, en esta versión
⁹¹ se cambiaron los GLMs por GAMS. Estos nuevos modelos están siendo muy utilizados en
⁹² diversas áreas porque heredan lo mejor de los GLM pero son mucho más flexibles. En la
⁹³ siguiente sección se describirán brevemente los GAMS.

⁹⁴ 4.1 Introducción a los Modelos Aditivos Generalizados

⁹⁵ 4.1.1 Interpretabilidad vs Complejidad

⁹⁶ La modelación estadística es una tarea muy compleja y que, en muchos casos, demanda
⁹⁷ un compromiso entre la interpretabilidad y complejidad de los modelos. La siguiente Figura
⁹⁸ muestra tres alternativas con creciente nivel de complejidad.



Figure 6: Modelos estadísticos

⁹⁹ En el extremo izquierdo, con los modelos lineales:

- 100 • Es posible crear modelos ajustados a los datos que son lineales y relativamente fáciles
 101 de interpretar y explicar.
- 102 • Existe una línea recta que representa la relación entre dos variables y atraviesa la nube
 103 de puntos.
- 104 • Es posible graficar la relación y hacer predicciones a partir del modelo ajustado.
- 105 Sin embargo, los modelos lineales no siempre representan bien las relaciones entre variables
 106 ya que las relaciones no siempre son lineales. Las predicciones no serían buenas a partir de
 107 estos modelos.
- 108 En el otro extremo del espectro hay toda una serie de modelos de tipo “caja negra” como
 109 las redes neuronales, random forest, árboles de regresión. Estos modelos son muy buenos
 110 para predecir pero son muy difíciles de interpretar y de entender qué está sucediendo en el
 111 sistema. Son muy útiles para clasificar pero no sirven para entender cómo una variable se
 112 relaciona con el producto del modelo.
- 113 Los GAMs proveen un interesante punto intermedio ya que se pueden ajustar relaciones
 114 complejas y no lineales e interacciones pero estos modelos son explícitos y se pueden observar
 115 las relaciones entre variables y entender porque se produce un resultado determinado.

116 **4.1.1.1 Relaciones no lineales** En general, las relaciones entre variables de la natu-
 117 raleza no son lineales y adquieren patrones muy complejos. En la Figura 4 se muestra un
 118 ejemplo de datos sintéticas para ejemplificar este concepto.

```
library(mgcv)

set.seed(2) ## simulate some data...

dat <- gamSim(1, n=400, dist="normal", scale=0, verbose=FALSE)

dat <- dat[,c("y", "x0", "x1", "x2", "x3")]

ggplot2::ggplot(dat, ggplot2::aes(y=y, x=x2)) +
```

```
ggplot2::geom_point() +
ggplot2::theme_minimal()
```

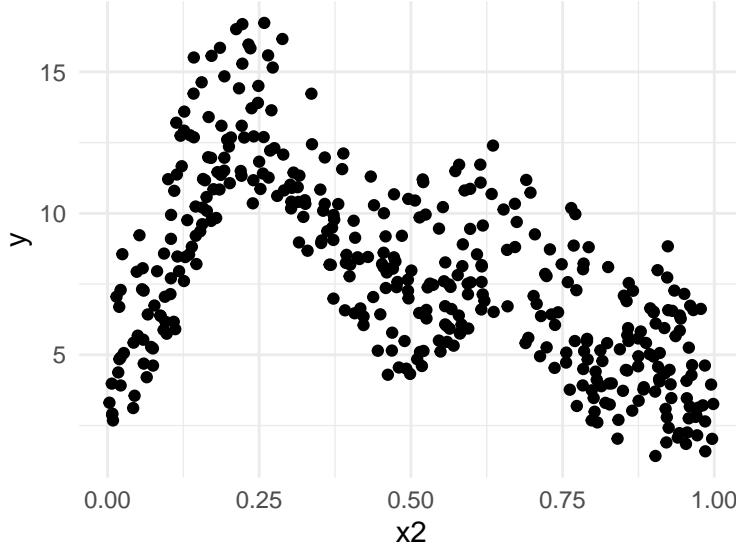


Figure 7: GAMs: Relaciones no lineales.

119 Este diagrama de dispersión muestra dos variables que claramente se encuentran relacionadas
 120 pero no de manera lineal. Si se ajusta una regresión lineal simple a estos datos se obtienen
 121 los siguientes resultados.

```
ggplot2::ggplot(dat, ggplot2::aes(x2, y)) +
ggplot2::geom_point() +
ggplot2::geom_smooth(method = 'lm', se = TRUE, ggplot2::aes(colour = "Lineal")) +
ggplot2::scale_colour_manual(name = "", values = c("Steelblue")) +
ggplot2::theme_minimal()
```

122 El modelo no es capaz de capturar las principales características de los datos.
 123 Modelo lineal

```
modelo_lineal <- lm(y ~ x2, data = dat)
```

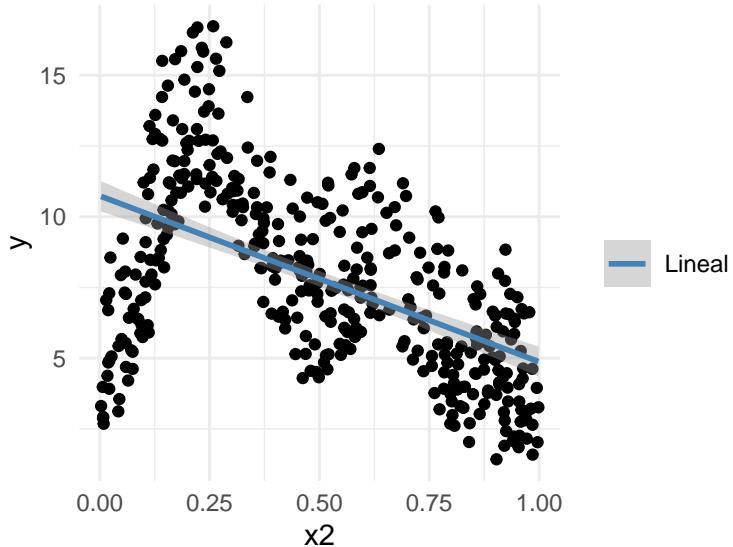


Figure 8: GAMs: Relaciones no lineales. Ajuste lineal.

124 Gráfico de residuos

```
residuals <- fortify(modelo_lineal)

ggplot2::ggplot(residuals, aes(x = .fitted, y = .resid)) +
  ggplot2::geom_point() +
  ggplot2::geom_smooth(se = FALSE) +
  ggplot2::geom_hline(yintercept = 0, linetype = 'dashed') +
  ggplot2::theme_bw() +
  ggplot2::xlab('Ajustados') + ggplot2::ylab('Residuos')

125 ## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

126 Claramente los residuos tienen una tendencia y no están distribuidos aleatoriamente.

127 Sin embargo, al cambiar el modelo lineal por un `gam`.

128 Con un GAM se pueden ajustar los modelos a través de funciones suavizadas o splines que
129 pueden tomar casi cualquier forma. Al utilizar splines los GAMs pueden capturar diversos
130 tipos de relaciones no lineales y es por esto que son tan flexibles y adaptables a distintos
131 contextos.

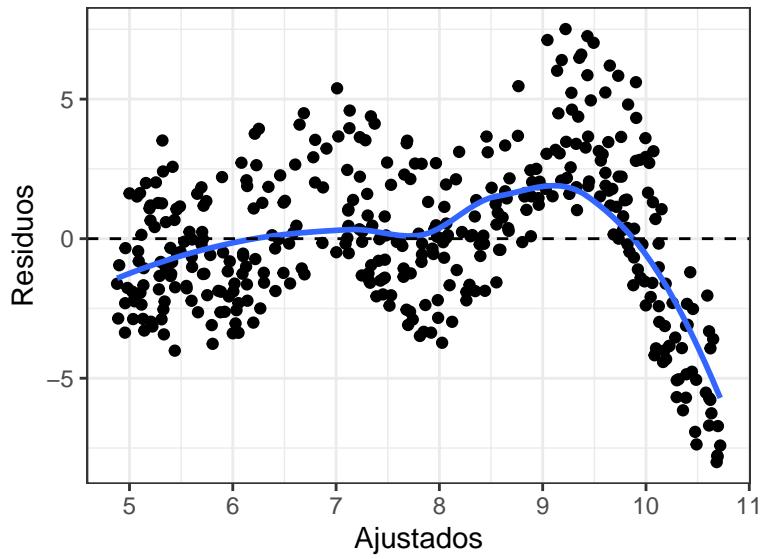


Figure 9: Residuos del modelo lineal.

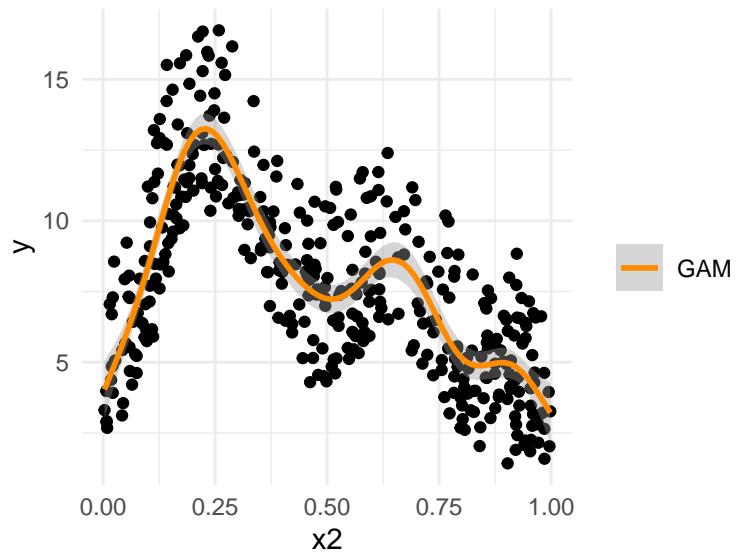


Figure 10: GAMs: Relaciones no lineales. Ajuste no lineal.

```

modelo_gam <- gam(y ~ s(x2), data = dat)

residuals <- data.frame(fitted = modelo_gam$fitted.values,
                        resid = resid(modelo_gam)) %>%
  tibble::as_tibble()

ggplot2::ggplot(residuals, aes(x = fitted, y = resid)) +
  ggplot2::geom_point() +
  ggplot2::geom_smooth(se = FALSE) +
  ggplot2::geom_hline(yintercept = 0, linetype = 'dashed') +
  ggplot2::theme_bw() +
  ggplot2::xlab('Ajustados') + ggplot2::ylab('Residuos')

132 ## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

```

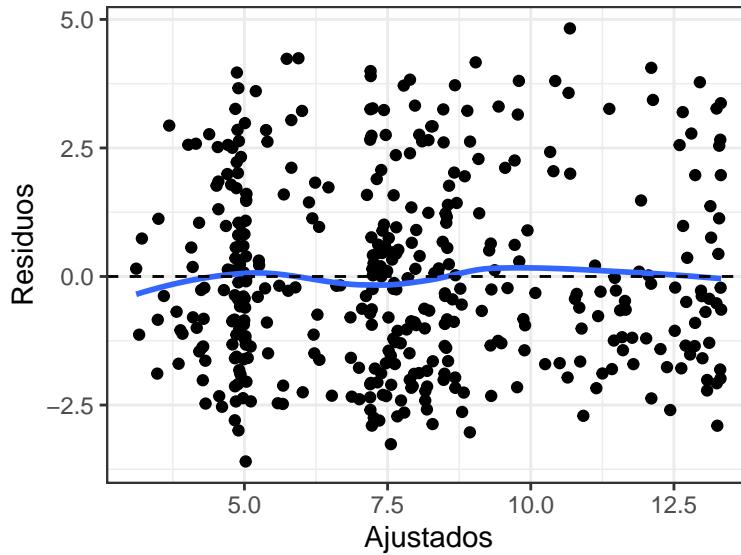


Figure 11: Residuos del GAM.

- 133 Al utilizar un GAM los residuos tienen una distribución casi aleatoria sin una tendencia
 134 clara como si fue el caso del modelo lineal. Este es sólo en sencillo ejemplo del potencial que

¹³⁵ tienen los GAMs para modelar complejas relaciones. Si se desea profundizar más en el tema
¹³⁶ se recomienda revisar el libro de Simon Wood (2017)

¹³⁷ 4.2 Modelación

¹³⁸ Como se mencionó anteriormente, el generador estocástico tiene como base cuatro modelos
¹³⁹ generalizados, dos para temperaturas máxima y mínima y dos para la precipitación que
¹⁴⁰ modelan la ocurrencia y los montos diarios.

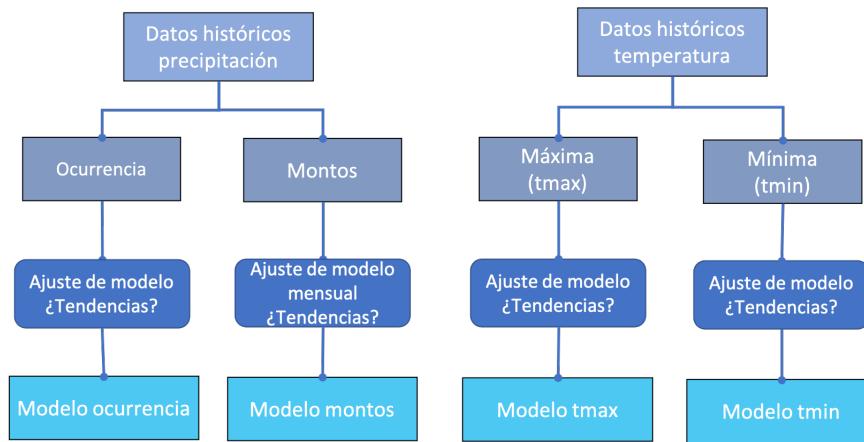


Figure 12: Modelos estadísticos

¹⁴¹ Partiendo de la base de datos se extraen los datos diarios de temperaturas máxima y mí-
¹⁴² nima y precipitación. Los valores de temperaturas son utilizados para ajustar un GAM que
¹⁴³ modelará la componente climática. Estos modelo puede incluir o no medias trimestrales que
¹⁴⁴ servirán para condicionar el modelo a seguir un determinado comportamiento. Para el caso
¹⁴⁵ de la precipitación el tratamiento es diferente. El fenómeno se divide en dos: ocurrencia de
¹⁴⁶ precipitación y montos diarios. La ocurrencia se modela con GAM para toda la serie que
¹⁴⁷ puede o no estar condicionado por totales trimestrales de lluvia. Los montos, en cambio, son
¹⁴⁸ modelados a escala mensual. Es decir, se ajustan doce modelos, uno para cada mes del año.
¹⁴⁹ Se utiliza esta modalidad para capturar mejor las características propias del ciclo estacional
¹⁵⁰ de la precipitación de cada región.

¹⁵¹ A continuación se describirán cada uno de los modelos recién mencionados.

152 **4.2.1 Clima local: Ocurrencia de lluvia**

153 El ajuste del modelo de ocurrencia comienza con la definición de un día lluvioso. Según la
154 Organización Meteorológica Mundial (OMM), se considera como día lluvioso a aquel día con
155 una precipitación igual o mayor a 0.1 mm. Este valor, si bien es el sugerido por la OMM,
156 puede ser modificado por el usuario si así lo dispone. Una vez definida la ocurrencia de lluvia
157 se ajusta el modelo. La ocurrencia de lluvia es una variable de tipo binaria, es decir, un día
158 puede ser lluvioso (1) o seco (0), y es muy bien representada a través de una regresión probit
159 Kleiber *et al.* (2012). Este tipo de regresiones se basan en procesos latentes Gaussianos,
160 $W_{s,t}$, que se modelan con la siguiente relación:

$$O_{s,t} = \Pi_{\{W_{s,t} > 0\}}$$

161 Si el proceso $W_{s,t}$ es positivo, significa que lloverá en el día t y en la estación s y a ese día
162 se le asignará el valor 1. Si el proceso es negativo significa que no lloverá en el día t y en la
163 estación s y ese día se le asignará el valor 0. El uso de este tipo de procesos latentes está
164 justificado en que, en una región, la ocurrencia de lluvia en las distintas estaciones tenderá a
165 estar correlacionada. La función media del proceso latente Gaussiano es una regresión entre
166 variables que se expresa de la siguiente manera:

$$O_{s,t} = (s, O_{s,t-1}, f(doy(t)), f(ST(t)), f(lon, lat))$$

167 donde $O_{s,t}$ corresponde a la ocurrencia de lluvia en sitio y día determinado; s corresponde
168 al efecto del sitio s (ordenada al origen); $O_{s,t-1}$ corresponde a la ocurrencia del día previo
169 que es un término autorregresivo; $f(doy(t))$ corresponde a una función cíclica de los días
170 del año para considerar el efecto de la estacionalidad sobre la ocurrencia de lluvia; $f(ST(t))$
171 corresponde a una función suavizada de los acumulados estacionales de precipitación (solo
172 se utiliza si se desea condicionar el modelo) y $f(lon, lat)$ corresponde a las coordenadas del

¹⁷³ punto **s** (solo se utiliza en el modelo espacial). En la práctica, esta covariable se divide en
¹⁷⁴ cuatro, una para cada trimestre del año, asignándole el valor de 0 para los momentos fuera del
¹⁷⁵ respectivo trimestre. Es importante notar que para la ocurrencia de precipitación se usa un
¹⁷⁶ solo término autorregresivo. Este término es muy importante para la correcta modelización
¹⁷⁷ de las rachas secas y lluviosas mientras que el día del año incorpora la variabilidad intra-
¹⁷⁸ anual.

¹⁷⁹ **4.2.2 Clima local: Montos diarios de lluvia**

¹⁸⁰ El modelo de montos diarios de lluvia difiere del anterior en que no se usan todos los datos
¹⁸¹ de la serie, sino que se extraen de la base de datos los montos de precipitación únicamente
¹⁸² para los días lluviosos. La intensidad de la precipitación para una localidad **s** y tiempo **t** es
¹⁸³ modelada como una variable aleatoria Gamma cuyos parámetros de forma y escala varían
¹⁸⁴ en el tiempo y en el espacio (Kleiber *et al.* (2012)). La función Gamma ha sido ampliamente
¹⁸⁵ utilizada en la región para la modelación de acumulados de lluvia. El modelo de montos
¹⁸⁶ puede ser expresado de la siguiente manera:

$$I_{s,t} = (s, O_{s,t-1}, f(ST(t)), f(lon, lat))$$

¹⁸⁷ donde, $I_{s,t}$ corresponde a los montos de precipitación en un sitio y día determinado; $O_{s,t-1}$
¹⁸⁸ corresponde a la ocurrencia del día previo para considerar la autocorrelación temporal;
¹⁸⁹ $f(ST(t))$ corresponde a una función suavizada de los acumulados estacionales de precipi-
¹⁹⁰ tación (solo se utiliza si se desea condicionar el modelo) y $f(lon, lat)$ corresponde a las
¹⁹¹ coordenadas del punto **s** (solo se utiliza en el modelo espacial). A diferencia del modelo
¹⁹² anterior – que modela los años calendarios completos a través de la estacionalidad del día
¹⁹³ del año – la serie de montos diarios es dividida en función del mes del año para ajustar una
¹⁹⁴ distribución a cada mes para obtener así parámetros mensuales más precisos. Gracias a esta
¹⁹⁵ modificación se obtienen parámetros que varían en el tiempo y en el espacio lo que permite

196 capturar la variabilidad espacial de la precipitación. Al igual que en el modelo de ocurrencia,
197 la inclusión del total trimestral es necesaria si se desean simular series condicionadas.

198 **4.2.3 Clima local: Temperatura**

199 Para el caso de la temperatura se utiliza una metodología similar a la utilizada para la precipi-
200 tación, basada en Kleiber *et al.* (2013). A partir de los datos observados de temperatura se
201 ajustan dos modelos: uno de máxima y otro de mínima. Ambos modelos utilizan las mismas
202 variables para realizar el ajuste. El modelo puede ser expresado de la siguiente manera:

$$X_{s,t} = (s, O_{s,t}, O_{s,t-1}, f(T_{x_{(s,t-1)}}, T_{n_{(s,t-1)}}), f(doy(t)), f(SX(t), SN(t)), f(lon, lat))$$

203 donde, $X_{s,t}$ corresponde a la temperatura máxima o mínima en un sitio y momento de-
204 terminado; s corresponde al efecto del sitio s (ordenada al origen); $O_{s,t}$ corresponde a la
205 ocurrencia de lluvia en sitio y día determinado; $O_{s,t-1}$ corresponde a la ocurrencia del día
206 previo; $f(T_{x_{(s,t-1)}}, T_{n_{(s,t-1)}})$ corresponde a una función suavizada de la interacción entre la
207 temperatura máxima y mínima del día previo; $f(doy(t))$ corresponde a una función cíclica
208 de los días del año para considerar el efecto de la estacionalidad sobre la temperatura; El
209 término $f(SX(t), SN(t))$ corresponde a la interacción entre las medias estacionales de tem-
210 peratura máxima y mínima y, como se explicó en la sección anterior, se incluyen para crear
211 modelos condicionados y $f(lon, lat)$ corresponde a las coordenadas del punto s (solo se uti-
212 liza en el modelo espacial). La ocurrencia de lluvia es muy importante ya que en general
213 los días lluviosos tienen temperaturas más bajas que los secos, sobre todo en verano, por lo
214 que debe ser incluido en el ajuste. La Figura 6 es un ejemplo de esta influencia en Junín,
215 en donde se muestra la amplitud térmica para cada mes del año en función del tipo de día,
216 seco o lluvioso.

217 **4.2.4 Tiempo local: Para una estación**

218 **4.2.4.1 Precipitación** El **tiempo local** de la ocurrencia de precipitación se modela
219 a través de los residuos de la regresión probit. Por definición estos residuos tienen una
220 distribución $X \sim \mathcal{N}(0, 1)$. Por lo tanto, generar valores para el tiempo local es muy sencillo,
221 solo se necesita de una función gaussiana que genere números aleatorios. La Figura muestra
222 un ejemplo de la generación de clima local y tiempo local para una estación meteorológica
223 de Argentina.

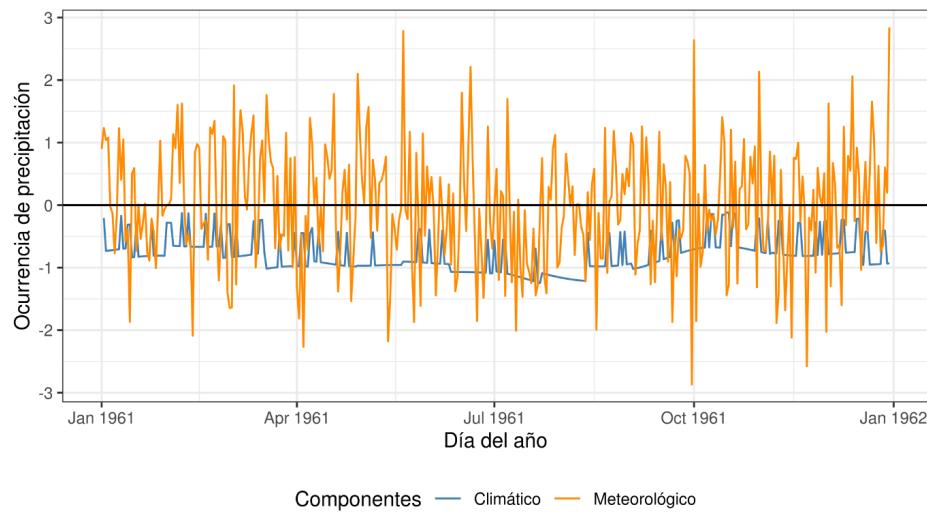


Figure 13: Tiempo local: precipitación

224 Esta Figura muestra sólo un año para mejorar la visualización pero su interpretación es válida
225 para toda la longitud de la serie. La línea azul corresponde al clima local modelado a través
226 del GAM y la naranja al ruido aleatorio creado a partir de una distribución $X \sim \mathcal{N}(0, 1)$.
227 La sumatoria de ambos componentes determinarán si el día es lluvioso o no. Si la suma es
228 positiva, lloverá, caso contrario será un día seco. Se puede observar en la línea azul un patrón
229 estacional debido al régimen de precipitación tipo monzónico de esta región con picos de más
230 días lluviosos durante el verano mientras que en invierno disminuyen marcadamente. Esta
231 misma serie temporal de números aleatorios serán la base para la generación de los montos
232 de precipitación.

233 **4.2.4.2 Temperatura** El tiempo local de las temperaturas máxima y mínima se modela
 234 de una manera diferente. En este caso se toman los residuos de cada uno de los GAMs, es
 235 decir, la diferencia entre el valor ajustado por el modelo y el valor observado de temperatura.
 236 Además, como la temperatura está fuertemente influenciada por el tipo de día, días lluviosos
 237 tienden a tener una menor amplitud térmica que los días secos, los residuos se separan en
 238 función del tipo de día. Para capturar mejor el patrón estacional de la temperatura, los
 239 residuos se agrupan por mes y se ajusta un modelo bivariado que contemple los residuos de
 240 temperaturas máxima y mínima. El uso de un modelos bivariado es una alternativa muy
 241 interesante para mantener la consistencia entre ambas variables, es decir, que la temperatura
 242 máxima no supere a la mínima. La siguiente Figura es un ejemplo de las series de tiempo
 243 local para una localidad de Argentina.

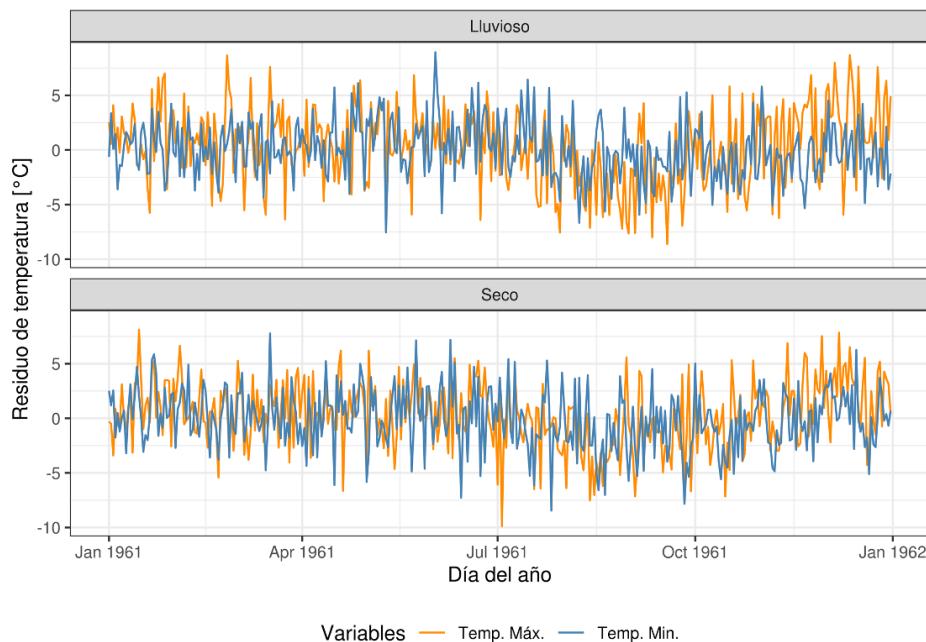


Figure 14: Tiempo local: temperatura

244 La figura esta dividida en dos paneles, el superior muestra el tiempo local para los días
 245 lluviosos y el inferior para los días secos. La línea naranja corresponde a los valores de
 246 tiempo local para temperatura máxima y los azules para temperatura mínima. Si bien
 247 ambas series difieren en magnitud y variabilidad, las dos tienden a variar conjuntamente.

²⁴⁸ **4.2.5 Tiempo local: Para una grilla**

²⁴⁹ Para generar datos sobre una grilla regular o en puntos donde no hay datos para ajustar los
²⁵⁰ modelos se debe utilizar el modelo espacial. La generación del tiempo local en el espacio
²⁵¹ es conceptualmente idéntico a la generación sobre estaciones meteorológicas sólo que utiliza
²⁵² campos gaussianos para incluir la dimensión espacial. Los campos gaussianos se generan con
²⁵³ el paquete `RandomFields` y capturan la variabilidad espacial de cada una de las variables a
²⁵⁴ partir de su variograma.

²⁵⁵ **4.2.5.1 Precipitación** Para la precipitación se utiliza un modelo exponencial que utiliza
²⁵⁶ como parámetros el variograma ajustado a partir de los datos observados usando máxima
²⁵⁷ verosimilitud. Este modelo permite simular campos con una muy buena consistencia espacial.
²⁵⁸ La ocurrencia de precipitación no ocurre de manera aislada en una región sino que, en general,
²⁵⁹ un evento lluvioso abarca una importante superficie. La siguiente Figura es un ejemplo de
²⁶⁰ los campos aleatorios generados sobre una grilla regular para una región de Argentina.

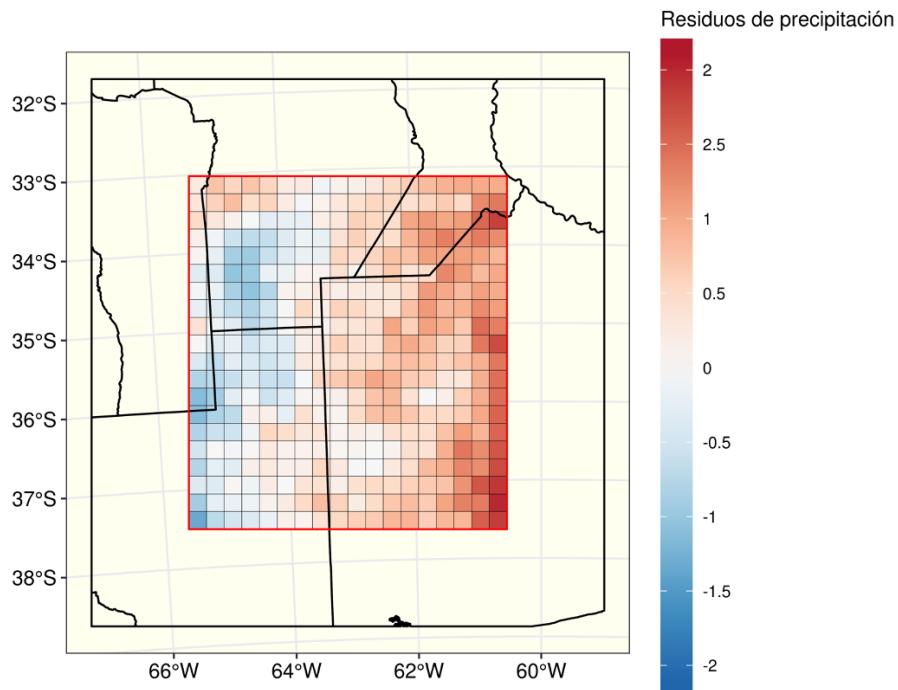


Figure 15: Tiempo local: precipitación sobre una grilla

261 La interpretación es análoga a la mostrada para una estación puntual. Los valores para cada
262 píxel se suman a la componente climática y así se determina si el día será lluvioso o no.
263 Este tipo de campos se generan para todos los días de la simulación y cada campo diario es
264 independiente del anterior.

265 **4.2.5.2 Temperatura** Al igual que para una estación, los campos gaussianos que se
266 generan son bivariados. Se utiliza el modelo Bivariado de Whittle Matern incluido en el
267 paquete `RandomFields`. La siguiente Figura es un ejemplo para un día de una realización
268 para grilla regular sobre Argentina.

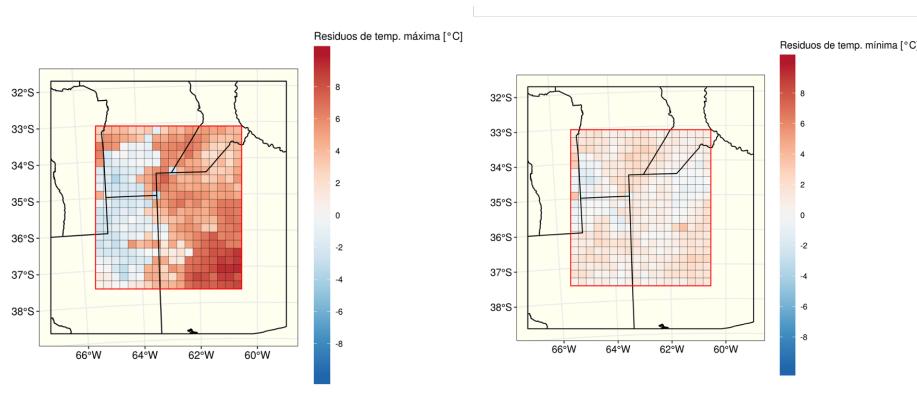


Figure 16: Tiempo local: temperatura sobre una grilla

269 Ambos campos se generan simultáneamente para temperaturas máxima y mínima y discrim-
270 inando entre días secos y lluviosos. Esto quiere decir que en el proceso de creación de los
271 campos se generan cuatro capas diferentes que luego se combinan en función del tipo de día
272 de cada píxel resultando en dos campos integradores. Los valores para cada píxel se sumen
273 a la componente climática y así se obtiene el valor final para cada día de la simulación.

274 5 Aplicación

275 En esta sección se mostrarán ejemplos de aplicación del generador para generar distintos
276 tipos de series explicando las funciones necesarias y cada uno de los parámetros.

277 5.1 Instalar paquetes necesarios

²⁷⁸ El primer paso es comprobar que todos los paquetes necesarios estén instalados y si no es así, descargarlos e instalarlos.

280 5.2 Creación de directorios

²⁸¹ El paquete tiene precargados algunos ejemplos de aplicación con datos reales de estaciones
²⁸² meteorológicas de la red del SISSA.

²⁸³ Para poder seguir este manual se deben crear directorios donde se guardarán los datos de
²⁸⁴ entrada y salida.

- 285 • /input_data: aquí se guardarán los datos meteorológicos y los metadatos de las esta-
286 ciones

287 • /output_data: aquí se guardarán los resultados de la simulación

288 Si estos directorios no existen, se crearán.

289 5.3 Generación de series sintéticas para una sola estación meteo- 290 rológica

Este primer ejemplo consiste en la generación de series sintéticas para estaciones meteorológicas. Es decir, se generarán series para las mismas estaciones que fueron utilizadas en el ajuste de los distintos modelos. Por lo tanto, no se considera la componente espacial.

²⁹⁴ El ejemplo está dividido en dos, en una primera parte se ajustará el modelo para generar
²⁹⁵ series estacionarias y luego, en una segunda parte, se incluirán covariables estacionales para
²⁹⁶ producir series pseudohistóricas. Para ambos ejemplos los datos son usados serán los mismos.

297 **5.3.1 Crear archivos de entrada**

298 El primer paso consiste en generar los set de datos de entrada que se descargaron al momento
299 de instalar el paquete del generador estocástico. Estos datos son sólo a título demostrativo,
300 si el usuario desea correr el modelo con sus propios datos deberá cambiar los objetos que se
301 generarán en esta sección por los suyos y colocarlos en la carpeta `input_data`.

302 Los archivos necesarios son:

- 303 • `stations.csv`
304 • `climate.csv`

305 Los datos meteorológicos se dividen en dos archivos separados: `stations.csv` y
306 `climate.csv`. Los nombres de los mismos no deben ser necesariamente iguales a los
307 usados aquí.

308 Los metadatos de las estaciones se alojan en el archivo `stations.csv`. Este archivo contiene
309 la información de las estaciones meteorológicas que serán usadas en el ajuste del modelo.

310 Las variables que deben ser incluidas en la tabla son:

- 311 • `station_id`: número único para cada estación meteorológica. La variable debe ser
312 de tipo *integer*
313 • `latitude`: latitud en grados decimales. La variable debe ser de tipo *double*
314 • `longitude`: longitud en grados decimales. La variable debe ser de tipo *double*

315 La tabla puede tener más variables pero sólo se necesitan las anteriores.

316 A continuación se muestran la primera fila del dataset y los tipo de datos de cada una de
317 las variables.

```
# Vista de los metadatos de la estación

knitr::kable(head(stations), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center")
```

x	y	station_id	nombre	lat_dec	lon_dec	elev	pais_id
5001614	6256841	87448	Villa Reynolds Aero	-33.7181	-65.3737	486	AR

318 El objeto **stations** debe ser convertido de *tibble* a *sf*. El sistema de referencia espacial debe
 319 ser planar. No es necesario un sistema de referencia espacial en particular, solamente las
 320 coordenadas deben estar expresadas en metros.

```
# Convertimos el objeto stations a sf y se convierte su proyección de WGS 1984 a
# POSGAR Argentina Faja 5.

stations %<>%
  sf::st_as_sf(coords = c("lon_dec", "lat_dec"), crs = 4326) %>%
  sf::st_transform(crs = 22185)
```

321 La información climática se aloja en el archivo `climate.csv`. Este archivo contiene los datos
 322 de las estaciones meteorológicas que serán usadas en el ajuste del modelo. Las variables que
 323 deben ser incluidas en la tabla son:

- 324 • `date`: fecha del dato. La variable debe ser de tipo *date*
- 325 • `station_id`: número único para cada estación meteorológica. La variable debe ser
 326 de tipo *integer*
- 327 • `prcp`: datos diarios de precipitación La variable debe ser de tipo *double*
- 328 • `tmax`: datos diarios de temperatura máxima. La variable debe ser de tipo *double*
- 329 • `tmin`: datos diarios de temperatura mínima. La variable debe ser de tipo *double*

330 A continuación se muestran las primeras cinco filas del dataset y los tipos de datos de cada
 331 una de las variables.

```
knitr::kable(head(climate), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center")
```

date	station_id	tmax	tmin	prcp
1961-01-01	87448	37.4	13.5	0.6
1961-01-02	87448	27.4	14.3	23.9
1961-01-03	87448	26.6	13.5	0.0
1961-01-04	87448	31.0	11.7	6.0
1961-01-05	87448	27.0	14.1	0.0
1961-01-06	87448	26.3	11.3	0.0

³³² Los nombres de las variables son importantes y deben ser siempre los mismos ya que el
³³³ modelo las reconocerá a partir de los mismos. Los nombres deben ser los siguientes:

³³⁴ • **date** : corresponde a la fecha del día en formato Date. El formato de la fecha para fa-
³³⁵ cilitar el reconocimiento por parte de R es “YYYY-MM-DD”, es decir, el año expresado
³³⁶ con cuatro dígitos y luego dos dígitos para el mes y dos para el día.

³³⁷ • **station_id**: Identificador único de cada una de las estaciones. Debe ser un número
³³⁸ entero.

³³⁹ • **tmax**: temperatura máxima diaria expresada en °C.

³⁴⁰ • **tmin**: temperatura mínima diaria expresada en °C.

³⁴¹ • **prcp**: precipitación diaria expresada en mm.

³⁴² El orden de las variables no es importante pero, como se mencionó, si se deben respetar los
³⁴³ nombres de cada una. En el caso de faltantes, no se utiliza ningún valor específico para los
³⁴⁴ NAs, sólo se debe dejar ese valor vacío. Este archivo tiene un formato largo, es decir, las
³⁴⁵ estaciones se deben colocar una debajo de la otra.

³⁴⁶ La generación de series sobre estaciones requiere de dos funciones básicas **local_fit** y
³⁴⁷ **local_simulate**. Independientemente de si se incluyen totales trimestrales en el modelo,
³⁴⁸ siempre se utilizan esas dos funciones.

349 **5.3.2 Series sintéticas estacionarias**

350 **5.3.2.1 Ajuste de los modelos** A continuación se mostrará como ajustar los cuatro
351 modelos estadísticos para una sola estación meteorológica para generar series estacionarias
352 donde cada realización es completamente independiente.

353 El ajuste del modelo local (en un punto) necesita de dos funciones: en una se define la
354 configuración general del modelo y con la segunda se corre el modelo propiamente dicho.

355 Primero se crea un objeto con el control para el ajuste del simulador. Los argumentos son:

- 356 • `prcp_occurrence_threshold`: umbral de precipitación para un día lluvioso. La OMM
357 recomienda un umbral de 0.1 mm para considerar un día como lluvioso.
- 358 • `avbl_cores`: cantidad de núcleos disponibles para la paralelización.
- 359 • `planar_crs_in_metric_coords`: sistema de coordenadas planar.

```
control_fit <- gamwgen::local_fit_control(  
  prcp_occurrence_threshold = 0.1, # Umbral para la definición de días húmedos  
  avbl_cores = 6, # Cantidad de núcleos disponibles  
  planar_crs_in_metric_coords = 22185) # Sistema de referencia espacial (en metros)
```

360 Luego se corre el ajuste para la estación meteorológica con la función `local_calibrate`.

361 Los argumentos de la función son:

- 362 • `climate`: datos meteorológicos observados para la estación
- 363 • `stations`: metadatos de las estaciones meteorológicas
- 364 • `seasonal_covariates`: datos agregados trimestrales. Si es NULL el ajuste será sin
365 covariables y las series generadas serán estacionarias.
- 366 • `control`: objeto de control
- 367 • `verbose`: controla la impresión de mensajes en la consola. FALSE por defecto.

```

# Al correr la función se realiza el ajuste de los cuatro modelos
# para cada una de las estaciones. En este caso, por cuestiones de tiempo
# a cargar un objeto ya precalculado.

# Si el usuario desea correrlo deberá ver la nota anterior.

gamgen_fit <- gamwgen::local_calibrate(
  climate = climate, # Registro histórico de variables meteorológicas
  stations = stations, # Estaciones meteorológicas
  seasonal_covariates = NULL, # Totales trimestrales de precipitación
  control = control_fit, # Objeto de control
  verbose = FALSE) # Impresión de mensajes en la consola.

```

368 Para esta demostración, cargamos el objeto con el ajuste del modelo ya realizado.

```

# Copiamos el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('input_data/local/fit_local_unconditional.RData')) {
  fs::file_copy(system.file('/autorun/local', "fit_local_unconditional.RData",
                           package = "gamwgen"),
               new_path = 'input_data/local/fit_local_unconditional.RData')
}

# Cargamos el archivo recientemente creado
load('input_data/local/fit_local_unconditional.RData')

# Clase del objeto con el ajuste del generador
class(gamgen_fit)

369 ## [1] "gamwgen"

```

```

# Contenido del modelo

names(gamgen_fit)

370 ## [1] "control"           "stations"          "climate"
371 ## [4] "crs_used_to_fit"    "start_climatology" "fitted_models"
372 ## [7] "models_data"        "models_residuals"   "statistics_threshold"
373 ## [10] "exec_times"

```

374 Dentro del objeto se guardan todo lo necesario para la simulación así como información
 375 accesoria.

- 376 • **control**: copia de la configuración usada para calibrar el generador
- 377 • **stations**: estaciones meteorológicas utilizadas para la calibración
- 378 • **climate**: datos climáticos de cada uno de las estaciones
- 379 • **seasonal_covariates**: series temporales de totales trimestrales de precipitación y
 380 medias trimestrales de temperaturas máxima y mínima.
- 381 • **crs_used_to_fit**: sistema de referencia espacial usado para proyectar
- 382 • **start_climatology**: climatología diaria de cada una de las variables de entrada.
- 383 • **fitted_models**: modelos ajustados, uno para cada variable: temperaturas máxima y
 384 mínima y ocurrencia y montos de precipitación.
- 385 • **models_data**: datos usados efectivamente usados para ajustar los modelos (sin NAs)
- 386 • **models_residuals**: residuos de cada uno de los modelos. Es decir, la diferencia entre
 387 el valor ajustado por el modelo (clima local) y el valor observado en el día **i**
- 388 • **statistics_threshold**: umbrales de amplitud térmica diaria por mes. Si la amplitud
 389 simulada está fuera de este rango, se repetirá la simulación para ese día a los fines de
 390 mantener la consistencia entre variables
- 391 • **exec_times**: tiempo de ejecución de cada una de las etapas del ajuste

392 Cada uno de los GAMs ajustados se almacenan en el objeto `gamgen_fit` y pueden ser
393 evaluados con la función `summary()`.

```
summary(gamgen_fit$fitted_models$`87448`$tmax_fit)

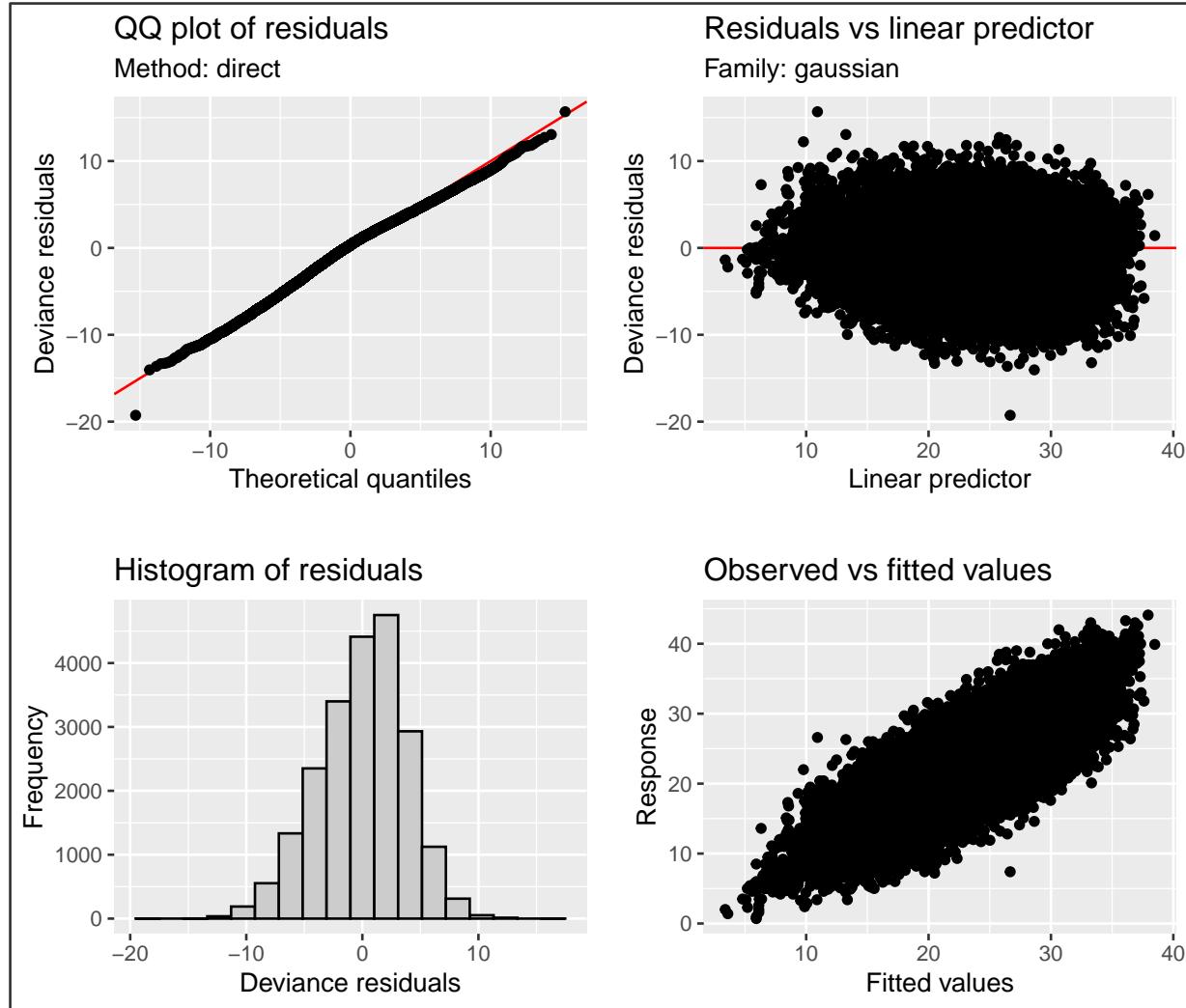
394 ##
395 ## Family: gaussian
396 ## Link function: identity
397 ##
398 ## Formula:
399 ## tmax ~ s(tmax_prev, tmin_prev, k = 50) + s(prcp_occ, bs = "re") +
400 ##       s(prcp_occ_prev, bs = "re") + s(doy, bs = "cc", k = 30)
401 ##
402 ## Parametric coefficients:
403 ##                               Estimate Std. Error t value Pr(>|t|)
404 ## (Intercept) 25.61343     0.03214    796.8 <2e-16 ***
405 ## ---
406 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
407 ##
408 ## Approximate significance of smooth terms:
409 ##                               edf Ref.df      F p-value
410 ## s(tmax_prev,tmin_prev) 30.2469  38.71 247.1 <2e-16 ***
411 ## s(prcp_occ)            0.9989   1.00 1005.8 <2e-16 ***
412 ## s(prcp_occ_prev)       0.9995   1.00 2300.2 <2e-16 ***
413 ## s(doy)                 12.8918  28.00 158.9 <2e-16 ***
414 ## ---
415 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
416 ##
```

```
417 ## R-sq.(adj) = 0.71 Deviance explained = 71.1%
418 ## fREML = 58955 Scale est. = 14.138 n = 21466
```

419 La función `summary` permite analizar los resultados del ajuste de cada uno de los modelos.
420 Para el caso del modelo de temperatura máxima podemos ver la fórmula del GAM en la parte
421 superior bajo el apartado `Formula` y la significancia de cada uno de los términos del modelo
422 en la tabla inmediatamente inferior. Se puede observar que todos los términos son altamente
423 significativos. También se incluyen como pruebas de bondad del ajuste el porcentaje de la
424 varianza explicada por el modelo y el valor de R-ajustado.

425 El paquete `gratia` (Simpson, Gavin (2018)) es muy útil para la visualización de los ajustes
426 de manera gráfica. Esta función toma los diagnósticos por defecto de la función `gam.check`
427 del paquete `mgcv` (Wood, Simon (2001)) y los convierte en gráficos de la librería `ggplot2`
428 (Wickham, Hadley(2011)) que son visualmente muy atractivos. La figura está dividida en
429 cuatro paneles, cada uno mostrando un diagnóstico diferente. En el panel superior izquierdo
430 se muestra un Q-Q Plot de los residuos. Si el modelo ha ajustado satisfactoriamente los
431 datos, los puntos deberían estar sobre la recta 1:1 demarcada en rojo. El panel superior
432 derecho muestra los residuos vs el predictor lineal. El objetivo de este diagnóstico es que los
433 residuos se distribuyan al azar y que no tengan un patrón claro. La presencia de patrones
434 en los residuos indicaría que el modelo no ha explicado algún componente importante de
435 la variabilidad de los datos. En el panel inferior izquierdo se muestra un histograma de los
436 residuos siendo deseable que lo mismos tengan una distribución próxima a la Normal. El
437 último gráfico en el panel inferior derecho muestra una gráfica de dispersión entre los valores
438 ajustados y observados.

```
gratia::appraise(gamgen_fit$fitted_models`87448`$tmax_fit) +
  ggplot2::theme_bw()
```



439

440 Estos diagnósticos exploratorios pueden aplicarse a cada uno de los modelos ajustados por la
441 función `local_calibrate`.

442 Con la creación del objeto `gamgen_fit` finaliza el proceso de ajuste y ya se pueden generar
443 series sintéticas para la o las estaciones usadas para calibrar el generador.

444 **5.3.2.2 Generación de series** La generación de series sigue la misma estructura ante-
445 rior, una función para configurar la generación y otra que realiza la generación propiamente
446 dicha.

447 Los argumentos de la función de control son:

- **nsim**: cantidad de simulaciones a realizar. Se debe ingresar un valor **entero** mayor o igual a 1.
- **seed**: semilla. Se debe ingresar cualquier numero **entero**. No es necesario recordarlo porque se guarda junto a los resultados.
- **avbl_cores**: cantidad de núcleos disponibles para la paralelización.
- **use_spatially_correlated_noise**: utilizar la generación estocástica espacialmente correlacionada. Esta opción sólo es válida si en el ajuste y en la simulación se usaron más de cinco estaciones meteorológicas diferentes. Con un menor número no es posible calcular los variogramas necesarios para la generación de los campos aleatorios. Se debe introducir un **boolean** (TRUE or FALSE).
- **use_temporary_files_to_save_ram**: si se simulan muchas realizaciones o los recursos informáticos son escasos, esta opción permite guardar los resultados de cada una de las realizaciones en el disco liberando memoria RAM que quedará disponible para generar nuevas simulaciones. Al finalizar la generación todos los archivos se combinan en uno único. Se debe introducir un **boolean** (TRUE or FALSE).
- **use_temporary_files_to_save_ram**: esta opción permite eliminar los archivos temporales creados para ahorrar RAM luego de terminar la generación de todas las simulaciones. Se debe introducir un **boolean** (TRUE or FALSE).

```

control_sim <- gamwgen::local_simulation_control(
  nsim = 10, # Cantidad de simulaciones a realizar
  seed = 1234, # Semilla para que los resultados sean reproducibles
  avbl_cores = 6, # Cantidad de núcleos disponibles a utilizar
  use_spatially_correlated_noise = FALSE,
  # Usar modelo de ruido espacialmente correlacionado
  use_temporary_files_to_save_ram = FALSE,
  # Guardar resultados intermedios para ahorrar RAM
  remove_temp_files_used_to_save_ram = TRUE)

```

```
# Borrar los resultados intermedios creados anteriormente
```

466 Luego se procede a la simulación de datos meteorológicos. Los argumentos de la función de
467 simulación son:

- 468 • **model**: objeto con el resultado de la función `local_calibrate()`
- 469 • **simulation_locations**: objeto tipo `sf` con la ubicación de las estaciones a simu-
470 lar. Las estaciones usadas deben haber sido incluidas en el proceso de ajuste. No es
471 necesario que todas estén presentes, se pueden generar series solo sobre algunas de
472 ellas.
- 473 • **start_date**: fecha de comienzo de la generación de series sintéticas. Si no se incluyeron
474 covariables estacionales en el ajuste, la fecha de comienzo es completamente arbitraria.
475 Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de
476 covariables, ni tampoco posterior. Se debe introducir una fecha en formato **date**
- 477 • **end_date**: fecha de fin de la generación de series sintéticas. Si no se incluyeron co-
478 variables estacionales en el ajuste, la fecha de fin es completamente arbitraria. Caso
479 contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables,
480 tampoco puede ser posterior. Se debe introducir una fecha en formato **date**
- 481 • **control**: objeto de control creado con la función `control_sim()`.
- 482 • **output_folder**: ruta al directorio donde se guardarán los resultados, tanto finales
483 como intermedios.
- 484 • **output_filename**: nombre del archivo de salida. Para facilitar la interoperabilidad,
485 el archivo generado es un archivo de texto en formato separado por comas (.csv)
- 486 • **seasonal_covariates**: datos agregados trimestrales. Si el ajuste se realizó con co-
487 variables, la generación también debe realizarse con ellas. Caso contrario se producirá
488 un error. Se debe introducir un data frame con los valores agregados para las tres
489 variables (precipitación y temperaturas máxima y mínima) pero no necesariamente
490 deben ser los mismos a los utilizados en el ajuste. Si se desean simular tendencias de

algún tipo, ya sea de un modelo de cambio climático o arbitrarias, se deben perturbar estas variables trimestrales e introducirlas aquí. Estas series si deben tener la misma longitud que el período a generar

- **verbose**: controla la impresión de mensajes en la consola. FALSE por defecto.

```
# Al correr la función se realiza la generación de series para
# cada una de las estaciones.

# En este caso, por cuestiones de tiempo, vamos a cargar un objeto
# con los resultados de la simulación

simulated_climate <- gamwgen::local_simulation(
  model = gamgen_fit, # Objeto con los resultados del ajuste
  simulation_locations = stations,
  # Estaciones para las cuales simular
  start_date = as.Date('2019-01-01'),
  # Fecha de comienzo de las simulaciones
  end_date = as.Date('2019-12-31'),
  # Fecha de fin de las simulaciones
  control = control_sim,
  # Objeto con la configuración
  output_folder = getwd(),
  # Directorio donde se guardarán los resultados
  output_filename = 'simulations.csv',
  # Nombre del archivo de salida
  seasonal_covariates = NULL,
  # Covariables estacionales
  verbose = FALSE)

# Impresión de mensajes en la consola
```

- 495 Esta función produce dos tipos de resultados: una lista que permanece en el ambiente de R y
496 los datos generados que son guardados como .csv en el directorio indicado precedentemente.

```
# Copiamos el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('output_data/simulated_local_unconditional.RData')) {
  fs::file_copy(system.file('/autorun/local', "simulated_local_unconditional.RData",
                           package = "gamwgen"),
                new_path = 'output_data/simulated_local_unconditional.RData')
}

# Cargamos el archivo recientemente creado
load('output_data/simulated_local_unconditional.RData')

# Clase del objeto con el ajuste del generador
class(simulated_climate)

497 ## [1] "list"           "gamwgen.climate"

# Contenido del modelo
names(simulated_climate)

498 ## [1] "nsim"
499 ## [2] "seed"
500 ## [3] "realizations_seeds"
501 ## [4] "simulation_points"
502 ## [5] "output_file_with_results"
503 ## [6] "output_file_fomart"
504 ## [7] "rdata_file_with_fitted_stations_and_climate"
505 ## [8] "exec_times"
```

506 La lista contiene los siguientes objetos:

- 507 • `nsim`: cantidad de realizaciones.
- 508 • `seed`: semilla general para toda la generación. Corresponde a la que se incluye en la
509 función de control.
- 510 • `realization_seeds`: semillas para cada una de las realizaciones. Esto permite replicar
511 los resultados.
- 512 • `simulation_points`: puntos donde se generaron las series sintéticas.
- 513 • `output_file_with_results`: nombre del archivo con los resultados.
- 514 • `output_file_format`: tipo de archivo de salida, en este caso `.csv`.
- 515 • `rdata_file_with_fitted_stations_and_climate`: archivo `.RData` con los datos me-
516 teorológicos observados que fueron utilizados en el ajuste. También se incluyen los
517 metadatos de cada uno de esos puntos.
- 518 • `exec_times`: tiempo de ejecución de la generación.

519 Ahora veremos el formato del archivo de salida que contiene las series sintéticas.

520 Para desmenuzar la generación del tiempo local se pueden correr algunas de las funciones
521 que forman parte de `local_simulation`. Por ejemplo, del objeto `gamgen_fit` se extraen
522 los residuos y con la función `gamwgen:::generate_residuals_statistics` se calculan los
523 parámetros.

524 En la tabla anterior se muestran los parámetros necesarios para generar el tiempo local de
525 las temperaturas máxima y mínima.

```
gen_noise_params <- gamwgen:::generate_residuals_statistics(  
  models_residuals = gamgen_fit$models_residuals)
```

```
knitr::kable(head(gen_noise_params), "latex", booktabs = T) %>%
```

```
kableExtra::kable_styling(position = "center",
                           latex_options = c("striped", "scale_down"))
```

station_id	type_day	month	sd.tmax_residuals	sd.tmin_residuals	mean.tmax_residuals	mean.tmin_residuals	cov.residuals	var.tmax_residuals	var.tmin_residuals
87448	Wet	1	3.254469	2.543695	0.8526532	-0.3860514	1.0779804	12.389767	4.907709
87448	Dry	1	3.254469	2.543695	-0.3494070	0.2137263	1.6070642	9.379906	7.045536
87448	Wet	2	3.352673	2.589353	0.7087247	-0.2850597	0.8544476	13.069030	4.397535
87448	Dry	2	3.352673	2.589353	-0.3295625	0.0035795	1.5816505	10.215926	7.603715
87448	Dry	3	3.444108	2.678806	0.0577372	0.0427601	1.6112722	10.910244	7.743939
87448	Wet	3	3.444108	2.678806	-0.2366143	-0.0993789	1.8594848	14.323711	5.682934

- **station_id**: número único que identifica a cada estación meteorológica.
- **type**: tipo de día **lluvioso (Wet)** o **seco (Dry)**.
- **month**: número de mes para los que se calculan los parámetros
- **sd.tmax_residuals**: desvío estándar de los residuos del modelo de temperatura máxima.
- **sd.tmin_residuals**: desvío estándar de los residuos del modelo de temperatura mínima.
- **mean.tmax_residuals**: media de los residuos del modelo de temperatura máxima.
- **mean.tmin_residuals**: media de los residuos del modelo de temperatura mínima *
- **cov.residuals**: covarianza de los residuos.
- **var.tmax_residuals**: covarianza de los residuos del modelo de temperatura máxima.
- **var.tmin_residuals**: covarianza de los residuos del modelo de temperatura mínima.

Losa parámetros para cada uno de los meses permiten generar valores de tiempo local para las dos temperaturas a partir de una distribución normal multivariada.

A continuación se muestra un ejemplo para el año 2019 sólo para los días lluviosos.

```
fechas <- data.frame(
  date = seq(as.Date('2019-01-01'), as.Date('2019-12-31'), 'days')) %>%
  dplyr::mutate(dia = lubridate::day(date),
```

```

mes = lubridate::month(date))

# Ejemplo de generación de ruido para el mes de enero

tmax_dry <- purrr::map2_dfr(
  .x = fechas$dia,
  .y = fechas$mes,
  .f = function(dia, mes) {

    result_tmax_dry <- control_sim$temperature_noise_generating_function(
      simulation_points = stations %>%
        dplyr::filter(., station_id == '87448'),
      gen_noise_params = gen_noise_params,
      month_number = mes,
      selector = 'tmax_dry',
      seed = NULL)

    result_tmax_dry <- result_tmax_dry %>%
      dplyr::mutate(dia = dia,
                    mes = mes)
  }
) %>%
  sf::st_set_geometry(NULL) %>%
  dplyr::mutate(date = as.Date(paste0('2019-', mes, '-1', dia))) %>%
  dplyr::select(-mes, -dia)

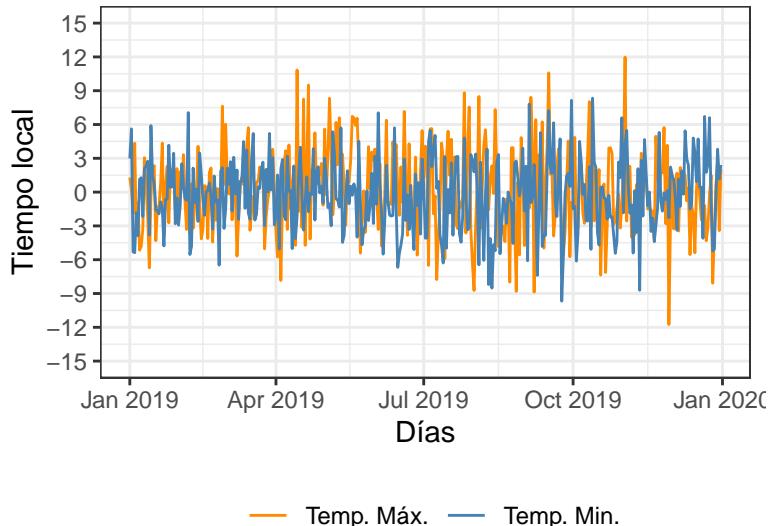
ggplot2::ggplot(data = tmax_dry %>%

```

```

tidy::gather(residuo, valor, -date),
ggplot2::aes(x = date, y = valor, color = residuo)) +
ggplot2::scale_y_continuous(limits = c(-15, 15),
                            breaks = seq(-15, 15, 3),
                            name = 'Tiempo local') +
ggplot2::scale_x_date(name = 'Días') +
ggplot2::scale_color_manual(values=c("DarkOrange", "Steelblue"),
                            labels = c("Temp. Máx.", "Temp. Min.)) +
ggplot2::geom_line() +
ggplot2::theme_bw() +
ggplot2::theme(legend.position="bottom",
              legend.title = ggplot2::element_blank())

```



541

- 542 La línea naranja corresponde a la serie para temperaturas máximas y la azul para temper-
 543 aturas mínimas.

```

# Se carga el set de datos simulados
simulated_climate <- readr::read_csv(

```

```

here::here('output_data/local/simulated_local_unconditional.csv'))

# Primeras filas del objeto de salidas

knitr::kable(head(simulated_climate), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center",
                            latex_options = c("striped", "scale_down"))

```

realization	station_id	point_id	longitude	latitude	date	tmax	tmin	prcp
1	87448	1	5001636	6256577	2019-01-01	27.19266	5.464899	0
1	87448	1	5001636	6256577	2019-01-02	30.57296	10.266542	0
1	87448	1	5001636	6256577	2019-01-03	33.36446	15.180860	0
1	87448	1	5001636	6256577	2019-01-04	34.05627	15.473379	0
1	87448	1	5001636	6256577	2019-01-05	32.69240	15.270145	0
1	87448	1	5001636	6256577	2019-01-06	30.56683	14.797530	0

544 El resultado de la generación es un archivo .csv que contiene la siguiente información:

- 545 • **realization**: número de realización. Es un valor entero entre 1 y la cantidad de
546 realizaciones definida por el usuario.
- 547 • **station_id**: número único de identificación de la estación meteorológica o del punto
548 arbitrario.
- 549 • **date**: fechas de cada uno de los días de la simulación.
- 550 • **tmax**: valores de temperatura máxima generada expresada en °C.
- 551 • **tmin**: valores de temperatura mínima generada expresada en °C.
- 552 • **prcp**: valores de precipitación diaria generada expresada en mm.

553 La siguiente Figura muestra un ejemplo de las series de temperaturas máximas y mínimas

554 generadas.

```

# Grafico de temperatura máxima

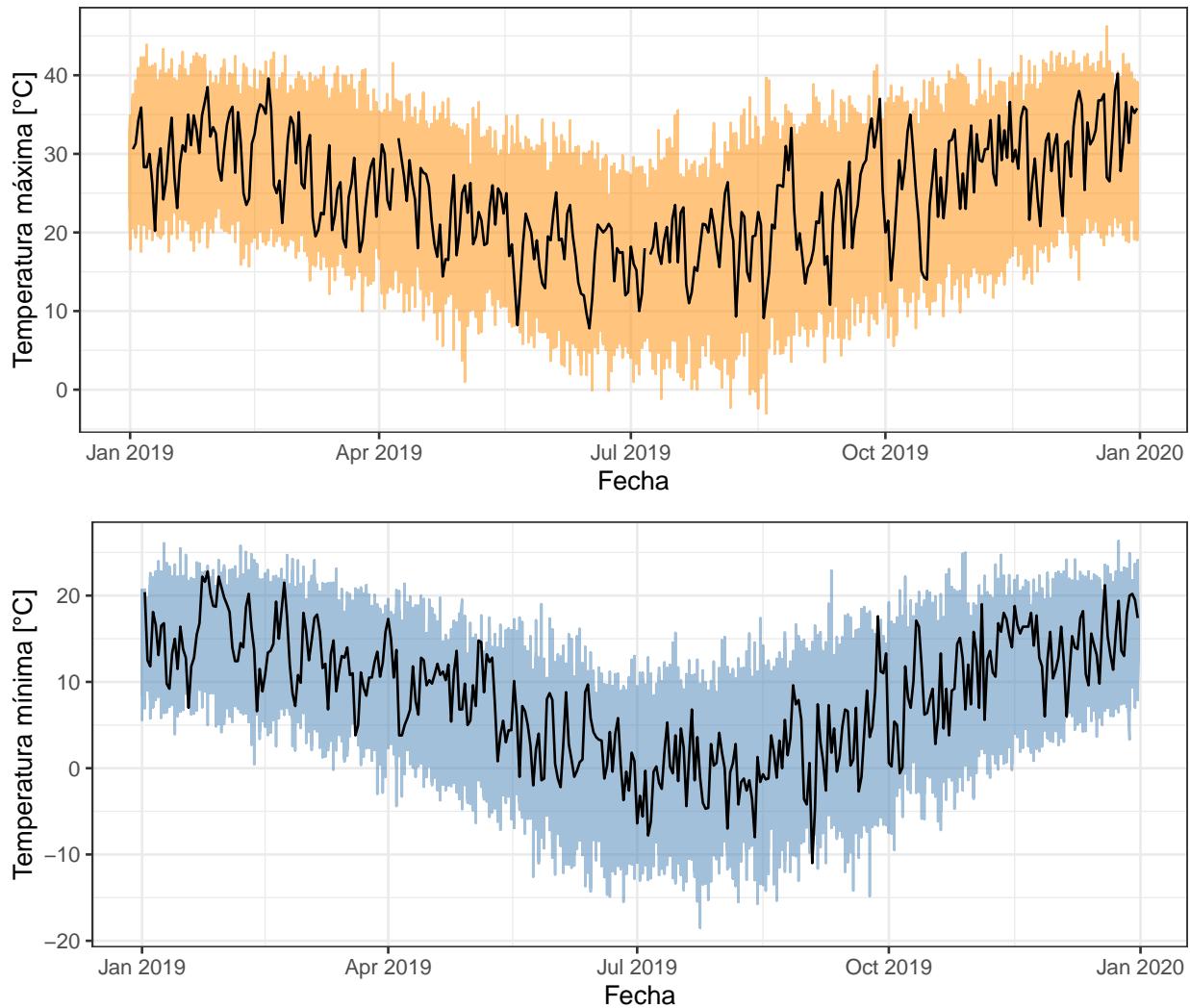
tmax_plot <- ggplot2::ggplot() +
  ggplot2::geom_line(data = simulated_climate,
    ggplot2::aes(x = date, y = tmax, color = realization),
    alpha = 0.5, color = 'DarkOrange') +
  ggplot2::geom_line(data = climate %>%
    dplyr::filter(date > as.Date('2019-01-01')),
    ggplot2::aes(x = date, y = tmax)) +
  ggplot2::labs(x = 'Fecha', y = 'Temperatura máxima [°C]') +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.position = "none")

# Grafico de temperatura mínima

tmin_plot <- ggplot2::ggplot() +
  ggplot2::geom_line(data = simulated_climate,
    ggplot2::aes(x = date, y = tmin, color = realization),
    alpha = 0.5, color = 'Steelblue') +
  ggplot2::geom_line(data = climate %>%
    dplyr::filter(date > as.Date('2019-01-01')),
    ggplot2::aes(x = date, y = tmin)) +
  ggplot2::labs(x = 'Fecha', y = 'Temperatura mínima [°C]') +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.position = "none")

cowplot::ggdraw() +
  cowplot::draw_plot(tmax_plot, x = 0, y = 0.5, width = 1, height = .5) +
  cowplot::draw_plot(tmin_plot, x = 0, y = 0, width = 1, height = .5)

```



555

556 En el panel superior se muestra la temperatura máxima observada (negra) y las temperaturas
 557 sintéticas (naranja) para el año 2019. En el inferior se muestra la temperatura mínima diaria
 558 observada (negra) y cada una de las realizaciones (azul).

559 **5.3.3 Series sintéticas pseudohistóricas**

560 **5.3.3.1 Ajuste de los modelos** A continuación se mostrará como ajustar los cuatro
 561 modelos estadísticos para una sola estación meteorológica para generar series **pseudo-**
 562 **históricas** donde cada realización copia las variaciones de baja frecuencia e la serie ob-
 563 servada. El ajuste del modelo local (en un punto) necesita de dos funciones: en una se define

564 la configuración general del modelo y con la segunda se corre el modelo propiamente dicho.

565 Primero se crea un objeto con el control para el ajuste del simulador. Los argumentos son:

- 566 • `prcp_occurrence_threshold`: umbral de precipitación para un día lluvioso. La OMM
567 recomienda un umbral de 0.1 mm para considerar un día como lluvioso.
- 568 • `avbl_cores`: cantidad de núcleos disponibles para la paralelización.
- 569 • `planar_crs_in_metric_coords`: sistema de coordenadas planar.

```
control_fit <- gamwgen::local_fit_control(  
  prcp_occurrence_threshold = 0.1, # Umbral para la definición de días húmedos  
  avbl_cores = 6, # Cantidad de núcleos disponibles  
  planar_crs_in_metric_coords = 22185) # Sistema de referencia espacial (en metros)
```

570 Al tratarse de un modelo que ajusta condicionado por la variabilidad de baja frecuencia
571 preexistente en los datos observados es necesario la agregación de las variables diarias en to-
572 tales trimestrales de precipitación y medias trimestrales de temperaturas máxima y mínima.
573 Esta operación puede realizarse con la función `summarise_seasonal_climate` incluida en
574 el paquete. Esta función, además de agregar los datos, permite la imputación de faltantes.
575 Se toleran una cierta cantidad que puede ser determinada por el usuario. El método de
576 imputación utilizado es el `imputePCA()` de la librería `missMDA`.

```
# Agregación de valores diarios  
  
seasonal_covariates <- gamwgen::summarise_seasonal_climate(climate, umbral_faltantes = 0)  
  
# Se muestran las primeras cinco filas  
  
knitr::kable(head(seasonal_covariates), "latex", booktabs = T) %>%  
  kableExtra::kable_styling(position = "center")
```

station_id	year	season	seasonal_prcp	seasonal_tmax	seasonal_tmin
87448	1961	1	273.7	30.98764	14.513483
87448	1961	2	236.0	24.47473	8.467033
87448	1961	3	11.3	19.04565	1.345652
87448	1961	4	234.5	24.66235	7.761177
87448	1962	1	402.4	29.92333	14.245556
87448	1962	2	187.2	24.30440	7.986813

⁵⁷⁷ Cabe mencionar que con esta función las valores se agregan por trimestre considerando la
⁵⁷⁸ siguiente definición:

- ⁵⁷⁹ • Verano: Diciembre, Enero y Febrero
⁵⁸⁰ • Otoño: Marzo, Abril y Mayo
⁵⁸¹ • Invierno: Junio, Julio y Agosto
⁵⁸² • Primavera: Septiembre, Octubre y Noviembre

⁵⁸³ Los valores también se podrían agregar siguiendo otra definición de estaciones pero en ese
⁵⁸⁴ caso, el usuario debería hacerlo por su cuenta. Algunas funciones útiles para hacerlo son
⁵⁸⁵ las disponibles en el paquete `lubridate` como `quarter()` que permite definir el mes de
⁵⁸⁶ comienzo de los trimestres. Para estas variables los nombres también son importantes por
⁵⁸⁷ lo que deben respetarse los mostrados anteriormente.

⁵⁸⁸ La siguiente Figura muestra los totales trimestrales de precipitación para la estación mete-
⁵⁸⁹ orológica del ejemplo.

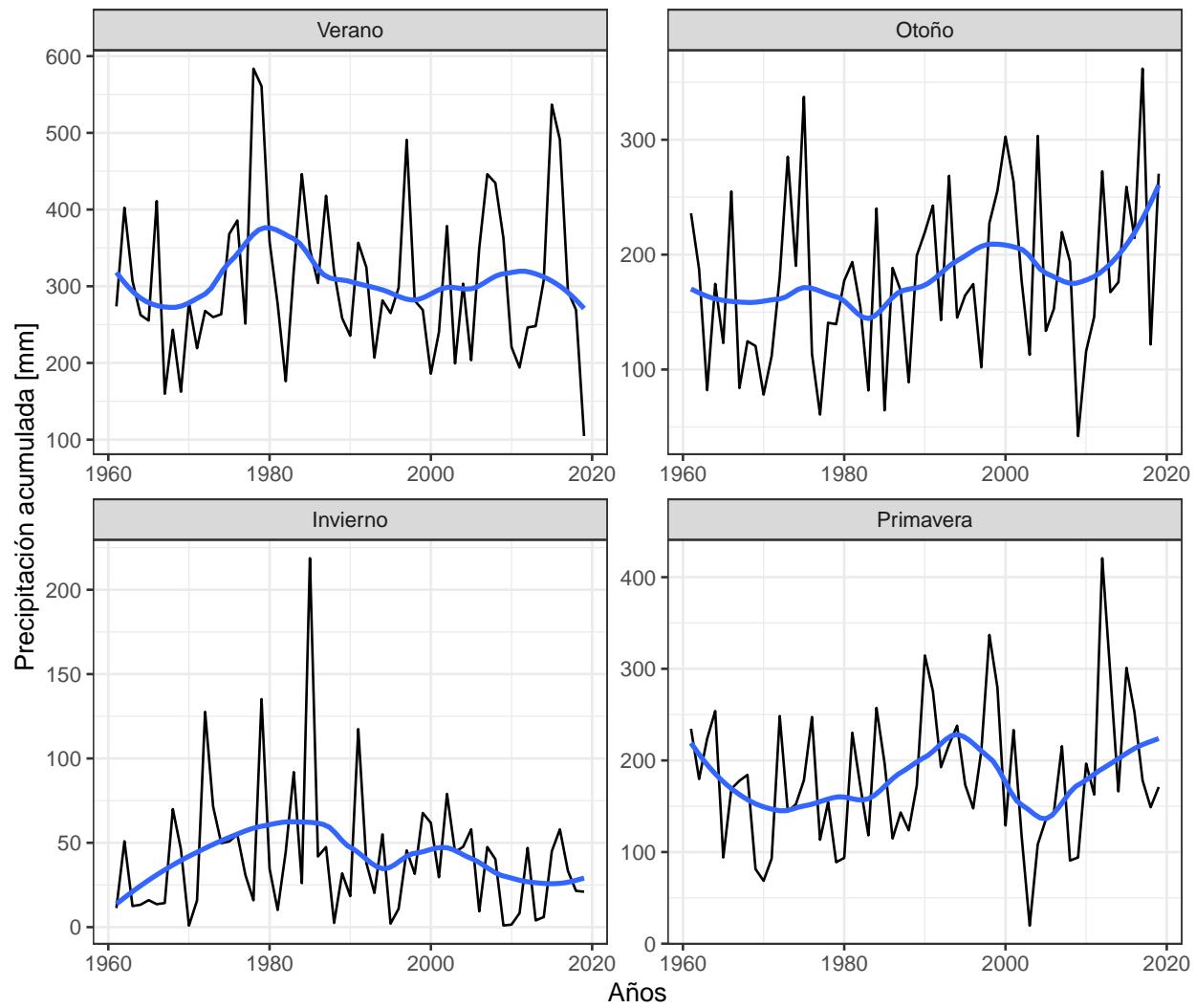
```
ggplot2::ggplot(data = seasonal_covariates %>%
  dplyr::mutate(season = factor(season,
  labels = c('Verano', 'Otoño', 'Inviern
```

```

ggplot2::aes(x = year, y = seasonal_prcp, group = 1)) +
  ggplot2::geom_line() +
  ggplot2::geom_smooth(se = FALSE, span = 0.5) +
  ggplot2::facet_wrap(~season, scales = 'free') +
  ggplot2::labs(x = 'Años', y = 'Precipitación acumulada [mm]') +
  ggplot2::theme_bw()

590 ## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

```



591

592 Cada uno de los paneles muestra la precipitación acumulada por trimestre junto a una
 593 regresión local (loess) en azul. Al incluir estos totales en el modelo, las series generadas

594 tenderán a seguir la variabilidad observada, es decir, años secos generarán realización con
595 valores inferiores a la media y viceversa.

596 Luego se corre el ajuste para la estación meteorológica con la función `local_calibrate`.

597 Los argumentos de la función son:

- 598 • `climate`: datos meteorológicos observados para la estación
- 599 • `stations`: metadatos de las estaciones meteorológicas
- 600 • `seasonal_covariates`: datos agregados trimestrales. Si es NULL el ajuste será sin
601 covariables y las series generadas serán estacionarias.
- 602 • `control`: objeto de control
- 603 • `verbose`: controla la impresión de mensajes en la consola. FALSE por defecto.

Al correr la función se realiza el ajuste de los cuatro modelos para
cada una de las estaciones. En este caso, por cuestiones de tiempo

a cargar un objeto ya precalculado.

Si el usuario desea correrlo deberá ver la nota anterior.

```
gamgen_fit <- gamwgen::local_calibrate(  
  climate = climate,  
  # Registro histórico de variables meteorológicas  
  stations = stations,  
  # Estaciones meteorológicas  
  seasonal_covariates = seasonal_covariates,  
  # Totales trimestrales de precipitación  
  control = control_fit,  
  # Objeto de control  
  verbose = FALSE)  
# Impresión de mensajes en la consola.
```

604 Para esta demostración, cargamos el objeto con el ajuste del modelo ya realizado.

```

# Copiamos el archivo preajustado a nuestro directorio de trabajo

if (!fs::file_exists('input_data/local/fit_local_conditional.RData')) {
  fs::file_copy(system.file('/autorun/local', "fit_local_conditional.RData",
                           package = "gamwgen"),
                new_path = 'input_data/local/fit_local_conditional.RData')
}

# Cargamos el archivo recientemente creado

load('input_data/local/fit_local_conditional.RData')

# Clase del objeto con el ajuste del generador

class(gamgen_fit)

605 ## [1] "gamwgen"

# Contenido del modelo

names(gamgen_fit)

606 ## [1] "control"           "stations"          "climate"
607 ## [4] "seasonal_covariates" "crs_used_to_fit"   "start_climatology"
608 ## [7] "fitted_models"        "models_data"       "models_residuals"
609 ## [10] "statistics_threshold" "exec_times"

```

610 Dentro del objeto se guardan todo lo necesario para la simulación así como información
 611 accesoria.

- 612 • **control**: copia de la configuración usada para calibrar el generador
- 613 • **stations**: estaciones meteorológicas utilizadas para la calibración
- 614 • **climate**: datos climáticos de cada uno de las estaciones

```

615 • seasonal_covariates: series temporales de totales trimestrales de precipitación y
616 medias trimestrales de temperaturas máxima y mínima.
617 • crs_used_to_fit: sistema de referencia espacial usado para proyectar
618 • start_climatology: climatología diaria de cada una de las variables de entrada.
619 • fitted_models: modelos ajustados, uno para cada variable: temperaturas máxima y
620 mínima y ocurrencia y montos de precipitación.
621 • models_data: datos usados efectivamente usados para ajustar los modelos (sin NAs)
622 • models_residuals: residuos de cada uno de los modelos. Es decir, la diferencia entre
623 el valor ajustado por el modelo (clima local) y el valor observado en el día i
624 • statistics_threshold: umbrales de amplitud térmica diaria por mes. Si la amplitud
625 simulada está fuera de este rango, se repetirá la simulación para ese día a los fines de
626 mantener la consistencia entre variables
627 • exec_times: tiempo de ejecución de cada una de las etapas del ajuste

628 Cada uno de los GAMs ajustados se almacenan en el objeto gamgen_fit y pueden ser
629 evaluados con la función summary().

```

```

summary(gamgen_fit$fitted_models$`87448`$tmax_fit)

630 ##
631 ## Family: gaussian
632 ## Link function: identity
633 ##
634 ## Formula:
635 ## tmax ~ s(tmax_prev, tmin_prev, k = 50) + s(prcp_occ, bs = "re") +
636 ##      s(prcp_occ_prev, bs = "re") + s(doy, bs = "cc", k = 30) +
637 ##      s(SX1, SN1, k = 20) + s(SX2, SN2, k = 20) + s(SX3, SN3, k = 20) +
638 ##      s(SX4, SN4, k = 20)

```

```

639  ##
640 ## Parametric coefficients:
641 ##             Estimate Std. Error t value Pr(>|t|)
642 ## (Intercept) 25.62116   0.03189   803.4 <2e-16 ***
643 ## ---
644 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
645 ##
646 ## Approximate significance of smooth terms:
647 ##             edf Ref.df      F p-value
648 ## s(tmax_prev,tmin_prev) 29.9069 38.405 225.74 <2e-16 ***
649 ## s(prcp_occ)           0.9989  1.000  930.30 <2e-16 ***
650 ## s(prcp_occ_prev)     0.9995  1.000 2248.11 <2e-16 ***
651 ## s(doy)                12.6336 28.000   73.12 <2e-16 ***
652 ## s(SX1,SN1)            3.0997  3.689   55.32 <2e-16 ***
653 ## s(SX2,SN2)            2.0001  2.000   89.25 <2e-16 ***
654 ## s(SX3,SN3)            4.4053  5.853   35.79 <2e-16 ***
655 ## s(SX4,SN4)            2.0001  2.000   86.85 <2e-16 ***
656 ## ---
657 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
658 ##
659 ## R-sq.(adj) =  0.713  Deviance explained = 71.4%
660 ## fREML = 58829  Scale est. = 13.964    n = 21466

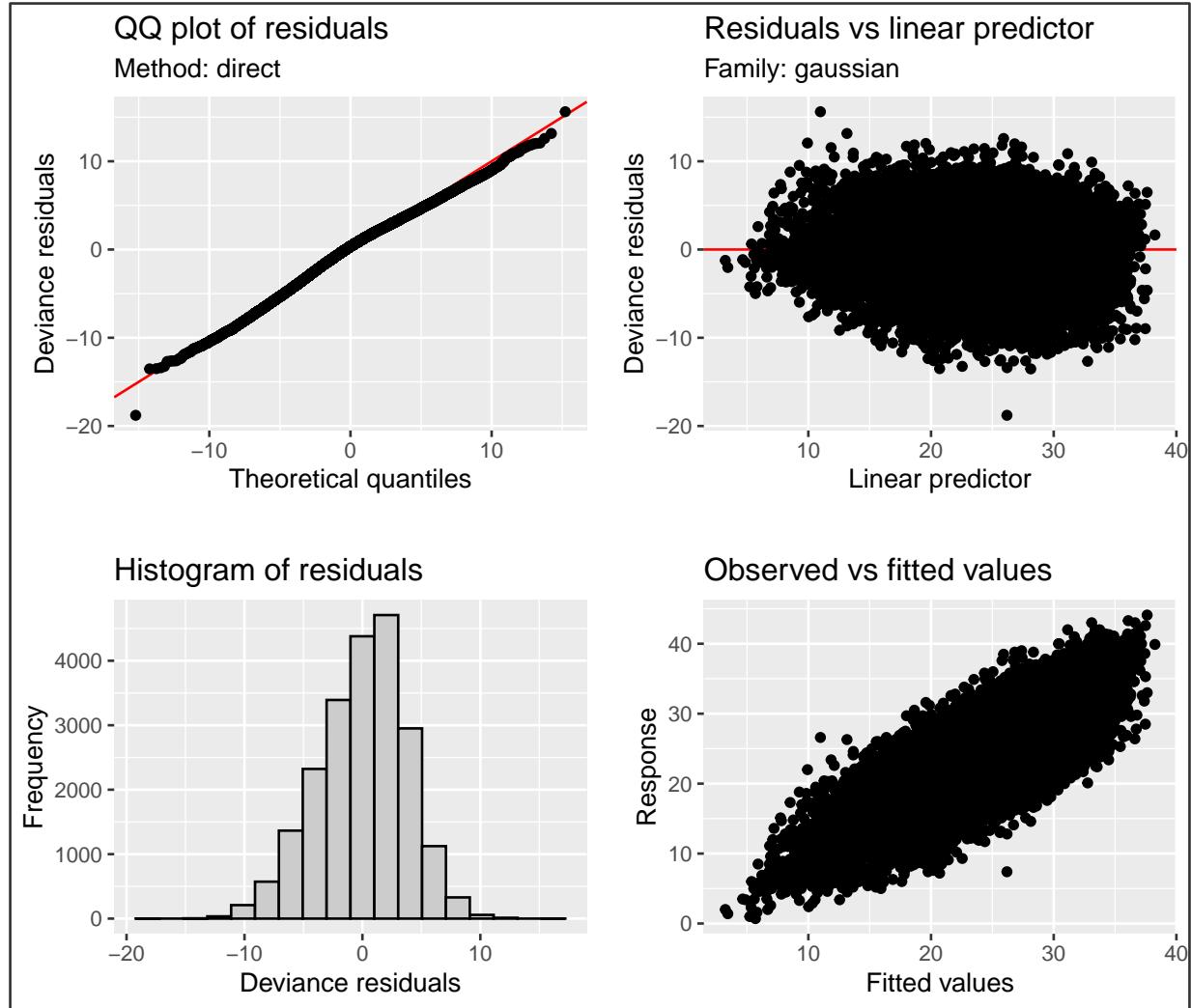
```

661 La función **summary** permite analizar los resultados del ajuste de cada uno de los modelos.
662 Para el caso del modelo de temperatura máxima podemos ver la fórmula del GAM en la
663 parte superior bajo el apartado **Formula** y la significancia de cada uno de los términos
664 del modelo en la tabla inmediatamente inferior. En este configuración, al incluirle variables
665 estacionales, se incorporan nuevos términos el modelo como SX1, SX2, SX3, SX4 y SN1, SN2,

666 SN3, SN4, que corresponden a variables dummy de las medias trimestrales de temperatura
667 máxima y mínima, respectivamente. Se puede observar que todos los términos son altamente
668 significativos. También se incluyen como pruebas de bondad del ajuste el porcentaje de la
669 varianza explicada por el modelo y el valor de R-ajustado.

670 El paquete **gratia** (Simpson, Gavin (2018)) es muy útil para la visualización de los ajustes
671 de manera gráfica. Esta función toma los diagnósticos por defecto de la función **gam.check**
672 del paquete **mgcv** (Wood, Simon (2001)) y los convierte en gráficos de la librería **ggplot2**
673 (Wickham, Hadley(2011)) que son visualmente muy atractivos. La figura está dividida en
674 cuatro paneles, cada uno mostrando un diagnóstico diferente. En el panel superior izquierdo
675 se muestra un Q-Q Plot de los residuos. Si el modelo ha ajustado satisfactoriamente los
676 datos, los puntos deberían estar sobre la recta 1:1 demarcada en rojo. El panel superior
677 derecho muestra los residuos vs el predictor lineal. El objetivo de este diagnóstico es que los
678 residuos se distribuyan al azar y que no tengan un patrón claro. La presencia de patrones
679 en los residuos indicaría que el modelo no ha explicado algún componente importante de
680 la variabilidad de los datos. En el panel inferior izquierdo se muestra un histograma de los
681 residuos siendo deseable que lo mismos tengan una distribución próxima a la Normal. El
682 último gráfico en el panel inferior derecho muestra una gráfica de dispersión entre los valores
683 ajustados y observados.

```
gratia::appraise(gamgen_fit$fitted_models`^87448`$tmax_fit) +  
  ggplot2::theme_bw()
```



684

685 Estos diagnósticos exploratorios pueden aplicarse a cada uno de los modelos ajustados por la
686 función `local_calibrate`.

687 Con la creación del objeto `gamgen_fit` finaliza el proceso de ajuste y ya se pueden generar
688 series sintéticas para la o las estaciones usadas para calibrar el generador.

689 **5.3.3.2 Generación de series** La generación de series sigue la misma estructura ante-
690 rior, una función para configurar la generación y otra que realiza la generación propiamente
691 dicha.

692 Los argumentos de la función de control son:

- **nsim**: cantidad de simulaciones a realizar. Se debe ingresar un valor **entero** mayor o igual a 1.
- **seed**: semilla. Se debe ingresar cualquier numero **entero**. No es necesario recordarlo porque se guarda junto a los resultados.
- **avbl_cores**: cantidad de núcleos disponibles para la paralelización.
- **use_spatially_correlated_noise**: utilizar la generación estocástica espacialmente correlacionada. Esta opción sólo es válida si en el ajuste y en la simulación se usaron más de cinco estaciones meteorológicas diferentes. Con un menor número no es posible calcular los variogramas necesarios para la generación de los campos aleatorios. Se debe introducir un **boolean** (TRUE or FALSE).
- **use_temporary_files_to_save_ram**: si se simulan muchas realizaciones o los recursos informáticos son escasos, esta opción permite guardar los resultados de cada una de las realizaciones en el disco liberando memoria RAM que quedará disponible para generar nuevas simulaciones. Al finalizar la generación todos los archivos se combinan en uno único. Se debe introducir un **boolean** (TRUE or FALSE).
- **use_temporary_files_to_save_ram**: esta opción permite eliminar los archivos temporales creados para ahorrar RAM luego de terminar la generación de todas las simulaciones. Se debe introducir un **boolean** (TRUE or FALSE).

```
control_sim <- gamwgen::local_simulation_control(
  nsim = 10, # Cantidad de simulaciones a realizar
  seed = 1234, # Semilla para que los resultados sean reproducibles
  avbl_cores = 6, # Cantidad de núcleos disponibles a utilizar
  use_spatially_correlated_noise = FALSE, # Usar modelo de ruido espacialmente correlacionado
  use_temporary_files_to_save_ram = FALSE, # Guardar resultados intermedios para ahorrar espacio
  remove_temp_files_used_to_save_ram = TRUE) # Borrar los resultados intermedios creados
```

711 Luego se procede a la simulación de datos meteorológicos. Los argumentos de la función de

712 simulación son:

- 713 • **model**: objeto con el resultado de la función `local_calibrate()`
- 714 • **simulation_locations**: objeto tipo `sf` con la ubicación de las estaciones a simular. Las estaciones usadas deben haber sido incluidas en el proceso de ajuste. No es necesario que todas estén presentes, se pueden generar series solo sobre algunas de ellas.
- 718 • **start_date**: fecha de comienzo de la generación de series sintéticas. Si no se incluyeron covariables estacionales en el ajuste, la fecha de comienzo es completamente arbitraria. Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables, ni tampoco posterior. Se debe introducir una fecha en formato `date`
- 722 • **end_date**: fecha de fin de la generación de series sintéticas. Si no se incluyeron covariables estacionales en el ajuste, la fecha de fin es completamente arbitraria. Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables, tampoco puede ser posterior. Se debe introducir una fecha en formato `date`
- 726 • **control**: objeto de control creado con la función `control_sim()`.
- 727 • **output_folder**: ruta al directorio donde se guardarán los resultados, tanto finales como intermedios.
- 729 • **output_filename**: nombre del archivo de salida. Para facilitar la interoperabilidad, el archivo generado es un archivo de texto en formato separado por comas (.csv)
- 731 • **seasonal_covariates**: datos agregados trimestrales. Si el ajuste se realizó con covariables, la generación también debe realizarse con ellas. Caso contrario se producirá un error. Se debe introducir un data frame con los valores agregados para las tres variables (precipitación y temperaturas máxima y mínima) pero no necesariamente deben ser los mismos a los utilizados en el ajuste. Si se desean simular tendencias de algún tipo, ya sea de un modelo de cambio climático o arbitrarias, se deben perturbar estas variables trimestrales e introducirlas aquí. Estas series si deben tener la misma longitud que el período a generar

- 739 • `verbose`: controla la impresión de mensajes en la consola. FALSE por defecto.

```
# Al correr la función se realiza la generación de series para cada una de las estaciones
# En este caso, por cuestiones de tiempo, vamos a cargar un objeto
# con los resultados de la simulación

simulated_climate <- gamwgen::local_simulation(
  model = gamgen_fit,
  # Objeto con los resultados del ajuste
  simulation_locations = stations,
  # Estaciones para las cuales simular
  start_date = as.Date('2010-01-01'),
  # Fecha de comienzo de las simulaciones
  end_date = as.Date('2019-12-31'),
  # Fecha de fin de las simulaciones
  control = control_sim,
  # Objeto con la configuración
  output_folder = getwd(),
  # Directorio donde se guardarán los resultados
  output_filename = 'simulations.csv',
  # Nombre del archivo de salida
  seasonal_covariates = seasonal_covariates,
  # Covariables estacionales
  verbose = FALSE)

# Impresión de mensajes en la consola
```

- 740 Esta función produce dos tipos de resultados: una lista que permanece en el ambiente de R y
741 los datos generados que son guardados como .csv en el directorio indicado precedentemente.

```
# Copiamos el archivo preajustado a nuestro directorio de trabajo
```

```

if (!fs::file_exists('output_data/local/simulated_local_conditional.RData')) {
  fs::file_copy(system.file('/autorun/local', "simulated_local_conditional.RData",
                           package = "gamwgen"),
               new_path = 'output_data/local/simulated_local_conditional.RData')
}

# Cargamos el archivo recientemente creado
load('output_data/local/simulated_local_conditional.RData')

# Clase del objeto con el ajuste del generador
class(simulated_climate)

742 ## [1] "list"           "gamwgen.climate"

# Contenido del modelo
names(simulated_climate)

743 ## [1] "nsim"
744 ## [2] "seed"
745 ## [3] "realizations_seeds"
746 ## [4] "simulation_points"
747 ## [5] "output_file_with_results"
748 ## [6] "output_file_fomart"
749 ## [7] "rdata_file_with_fitted_stations_and_climate"
750 ## [8] "exec_times"

751 La lista contiene los siguientes objetos:
    • nsim: cantidad de realizaciones.

```

- 753 • `seed`: semilla general para toda la generación. Corresponde a la que se incluye en la
 754 función de control.
- 755 • `realization_seeds`: semillas para cada una de las realizaciones. Esto permite replicar
 756 los resultados.
- 757 • `simulation_points`: puntos donde se generaron las series sintéticas.
- 758 • `output_file_with_results`: nombre del archivo con los resultados.
- 759 • `output_file_format`: tipo de archivo de salida, en este caso `.csv`.
- 760 • `rdata_file_with_fitted_stations_and_climate`: archivo `.RData` con los datos me-
 761 teorológicos observados que fueron utilizados en el ajuste. También se incluyen los
 762 metadatos de cada uno de esos puntos.
- 763 • `exec_times`: tiempo de ejecución de la generación.

764 Ahora veremos el formato del archivo de salida que contiene las series sintéticas.

```
# Se carga el set de datos simulados
simulated_climate <- readr::read_csv(
  here::here('output_data/local/simulated_local_conditional.csv'))

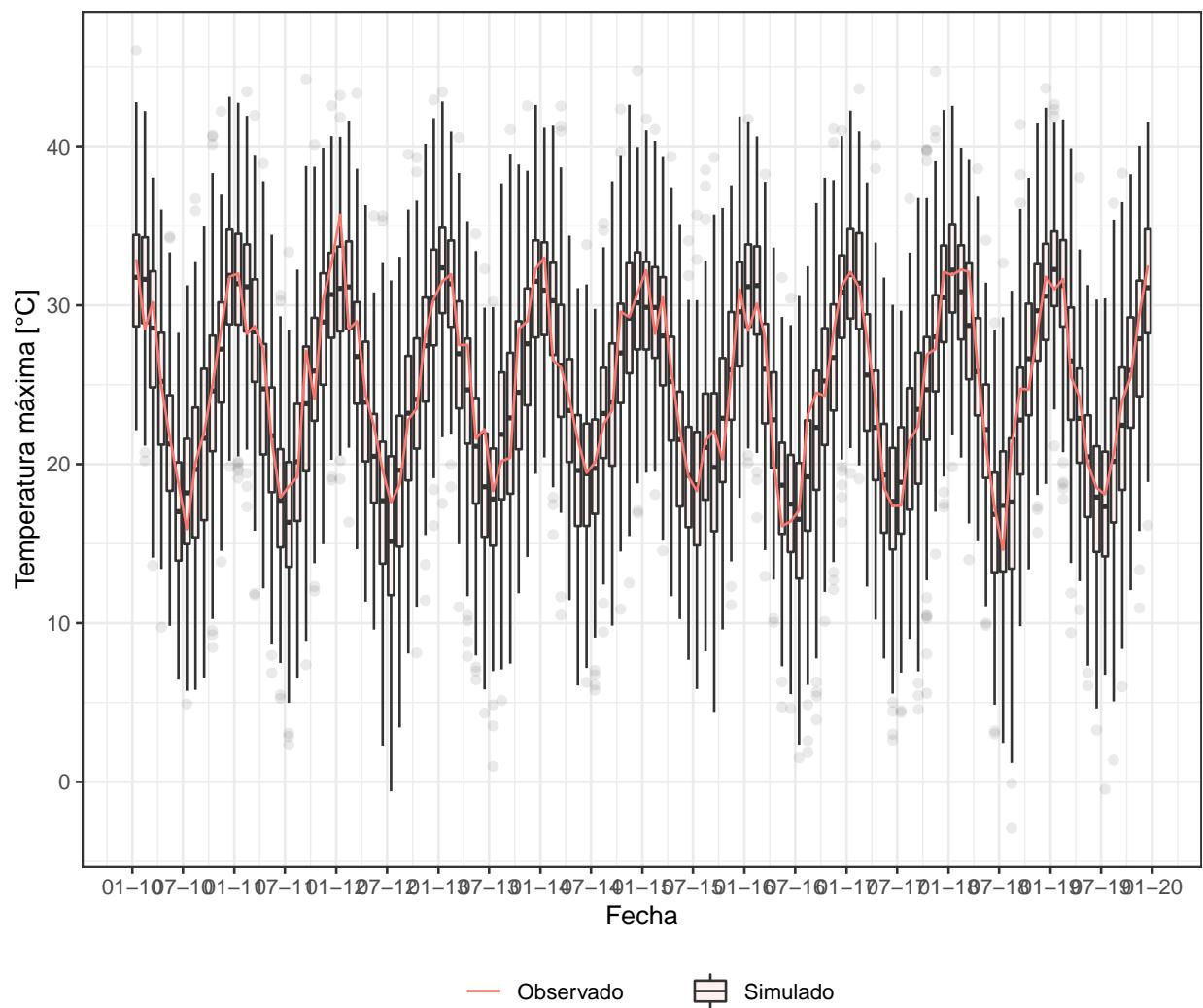
# Primeras filas del objeto de salidas
knitr::kable(head(simulated_climate), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center", latex_options = c("striped", "scale_down"))
```

realization	station_id	point_id	longitude	latitude	date	tmax	tmin	prcp
1	87448	1	5001636	6256577	2010-01-01	28.05965	17.69135	7.534622
1	87448	1	5001636	6256577	2010-01-02	27.32783	16.20019	0.000000
1	87448	1	5001636	6256577	2010-01-03	35.01025	13.35766	0.000000
1	87448	1	5001636	6256577	2010-01-04	34.18990	18.04710	0.000000
1	87448	1	5001636	6256577	2010-01-05	30.59661	20.62692	0.000000
1	87448	1	5001636	6256577	2010-01-06	37.15162	17.44571	0.000000

765 Al utilizar totales trimestrales de precipitación y medias trimestrales de temperaturas máx-
766 ima y mínima, las series generadas capturan los variaciones de baja frecuencia que se observan
767 en la serie histórica. A continuación se muestra una Figura que ilustra lo anterior para la
768 temperatura máxima.

```
ggplot2::ggplot() +  
  ggplot2::geom_boxplot(data = simulated_climate %>%  
    dplyr::mutate(month = lubridate::month(date),  
                 year = lubridate::year(date),  
                 date = as.Date(paste0(year, '-01', month, '01'))),  
    ggplot2::aes(x = date,  
                 y = tmax,  
                 group = date,  
                 fill = 'DarkOrange'),  
    alpha = 0.1) +  
  ggplot2::geom_line(data = climate %>%  
    dplyr::filter(date > as.Date('2010-01-01')) %>%  
    dplyr::mutate(month = lubridate::month(date),  
                 year = lubridate::year(date)) %>%  
    dplyr::group_by(year, month) %>%  
    dplyr::summarise(tmax = median(tmax, na.rm = TRUE)) %>%  
    dplyr::mutate(date = as.Date(paste0(year, '-01', month, '01'))),  
    ggplot2::aes(  
      x = date,  
      y = tmax,  
      group = 1,  
      color = 'DarkOrange')) +  
  ggplot2::theme_bw()
```

```
ggplot2::labs(x = 'Fecha', y = 'Temperatura máxima [°C]') +  
  ggplot2::scale_x_date(breaks = '6 months',  
                        labels = scales::date_format("%m-%y")) +  
  ggplot2::scale_fill_discrete(name = "", labels = c("Simulado")) +  
  ggplot2::scale_color_discrete(name = "", labels = c("Observado")) +  
  ggplot2::theme(legend.position = 'bottom')  
  
## `summarise()` regrouping output by 'year' (override with `groups` argument)
```



770

771 Las cajas corresponde a las distintas realizaciones agregadas a escala mensual y la línea
772 naranja corresponde a la temperatura media mensual calculada para los años 2010 a 2019.

⁷⁷³ Se observa como las cajas suben y bajan al ritmo de la media observada y como capturan
⁷⁷⁴ los pequeños cambios que ocurren en un año específico pero no en otros.

⁷⁷⁵ **5.3.4 Series sintéticas correlacionadas espacialmente**

⁷⁷⁶ Una tercera alternativa para la generación de datos sobre estaciones meteorológicas com-
⁷⁷⁷ bina las anteriores mostradas pero generando el tiempo local con métodos que contemplan la
⁷⁷⁸ autocorrelación espacial. Para utilizar esta alternativa se deben disponer de más de una
⁷⁷⁹ sola estación porque de otro modo no se podrían calcular los parámetros de los variogramas
⁷⁸⁰ necesarios para la generación del tiempo local. Mientras más estaciones haya mejor, pero si
⁷⁸¹ pueden generar campos espaciales confiables con alrededor de 10 puntos.

⁷⁸² **5.3.4.1 Crear archivos de entrada** Para este ejemplo se utilizan datos de varias esta-
⁷⁸³ ciones pero el formato de los mismos es igual a los mostrados anteriormente.

⁷⁸⁴ Los archivos necesarios son:

- ⁷⁸⁵ • stations.csv
⁷⁸⁶ • climate.csv

⁷⁸⁷ Los datos meteorológicos se dividen en dos archivos separados: **stations.csv** y
⁷⁸⁸ **climate.csv**. Los nombres de los mismos no deben ser necesariamente iguales a los
⁷⁸⁹ usados aquí.

⁷⁹⁰ Los metadatos de las estaciones se alojan en el archivo **stations.csv**. Este archivo contiene
⁷⁹¹ la información de las estaciones meteorológicas que serán usadas en el ajuste del modelo.

⁷⁹² Las variables que deben ser incluidas en la tabla son:

- ⁷⁹³ • station_id: número único para cada estación meteorológica. La variable debe ser
⁷⁹⁴ de tipo *integer*

- 795 • latitude: latitud en grados decimales. La variable debe ser de tipo *double*
 796 • longitude: longitud en grados decimales. La variable debe ser de tipo *double*

797 La tabla puede tener más variables pero sólo se necesitan las anteriores.

798 A continuación se muestran la primera fila del dataset y los tipo de datos de cada una de
 799 las variables.

Vista de los metadatos de la estación

```
knitr::kable(head(stations), "latex", booktabs = T) %>%  
  kableExtra::kable_styling(position = "center")
```

station_id	nombre	lat_dec	lon_dec	elev
86330	Artigas	-30.40	-56.51	120.38
86350	Rivera	-30.90	-55.54	241.94
86360	Salto	-31.43	-57.98	41.00
86430	Paysandú	-32.35	-58.04	61.12
86440	Melo	-32.37	-54.19	100.36
86460	Paso de los Toros	-32.80	-56.53	75.48

800 El objeto **stations** debe ser convertido de *tibble* a *sf*. El sistema de referencia espacial debe
 801 ser planar. No es necesario un sistema de referencia espacial en particular, solamente las
 802 coordenadas deben estar expresadas en metros.

Convertimos el objeto stations a sf y se convierte su proyección de WGS 1984 a
UTM Zona 21 S

```
stations %<>%  
  sf::st_as_sf(coords = c("lon_dec", "lat_dec"), crs = 4326) %>%  
  sf::st_transform(crs = 32721)
```

- 803 En el siguiente mapa muestra la distribución de las estaciones meteorológicas usadas en el
804 ejemplo.

```
# Convertir el objeto con las estaciones a coordenadas geográficas
stations_geo <- stations %>%
  sf:::st_transform(crs = 4326) %>%
  dplyr::mutate(lat = sf:::st_coordinates(.)[,2],
    lon = sf:::st_coordinates(.)[,1])

# Creación del mapa con la distribución de las estaciones
leaflet::leaflet(data = stations_geo) %>%
  leaflet::addTiles() %>%
  leaflet::setView(lat = -33, lng = -56, zoom = 5) %>%
  leaflet::addCircleMarkers(lat = ~lat, lng = ~lon, radius = 3,
    fillColor = "#377eb8",
    color = "#377eb8",
    stroke = FALSE, fillOpacity = 1, opacity = 1,
    popup = ~sprintf("<b>%s (%d)</b><br>Lat.: %.3f<br>Lon.: %.
      nombre, station_id, lat, lon, elev))
```



806 La información climática se aloja en el archivo `climate.csv`. Este archivo contiene los datos
 807 de las estaciones meteorológicas que serán usadas en el ajuste del modelo. Las variables que
 808 deben ser incluidas en la tabla son:

- 809 • `date`: fecha del dato. La variable debe ser de tipo `date`
- 810 • `station_id`: número único para cada estación meteorológica. La variable debe ser
 811 de tipo `integer`
- 812 • `prcp`: datos diarios de precipitación La variable debe ser de tipo `double`
- 813 • `tmax`: datos diarios de temperatura máxima. La variable debe ser de tipo `double`
- 814 • `tmin`: datos diarios de temperatura mínima. La variable debe ser de tipo `double`

815 A continuación se muestran las primeras cinco filas del dataset y los tipos de datos de cada
816 una de las variables.

```
knitr::kable(head(climate), "latex", booktabs = T) %>%  
  kableExtra::kable_styling(position = "center")
```

date	station_id	tmax	tmin	prcp
1981-01-01	86330	34.6	21.7	0.0
1981-01-02	86330	33.8	21.2	19.5
1981-01-03	86330	28.3	19.4	0.0
1981-01-04	86330	30.3	17.4	0.0
1981-01-05	86330	33.4	17.4	21.0
1981-01-06	86330	32.8	20.4	8.0

817 En total se utilizan 15 estaciones meteorológicas, 10 provistas por INUMET (Instituto
818 Uruguayo de Meteorología) y 5 por INIA (Instituto Nacional de Investigación Agropecuaria).

```
unique(climate$station_id)
```

```
819 ## [1] 86330 86350 86360 86430 86440 86460 86490 86560  
820 ## [9] 86565 86580 90000001 90000002 90000003 90000004 90000005
```

821 **5.3.4.2 Ajuste de los modelos** El ajuste de los modelos es igual que para los dos
822 ejemplos antes mostrados. Se utiliza la función `control_fit` para la configuración del ajuste
823 y `local_calibrate` para realizar el ajuste.

```
control_fit <- gamwgen::local_fit_control(  
  prcp_occurrence_threshold = 0.1, # Umbral para la definición de días húmedos
```

```

avbl_cores = 6, # Cantidad de núcleos disponibles
planar_crs_in_metric_coords = 32721) # Sistema de referencia espacial (en metros)

```

- 824 Este ejemplo se realizará utilizando covriables trimestrales pero es válido para series esta-
 825 cionarias.

Agregación de valores diarios

```

seasonal_covariates <- gamwgen::summarise_seasonal_climate(climate, umbral_faltantes =
  # Se muestran las primeras cinco filas
knitr::kable(head(seasonal_covariates), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center")

```

station_id	year	season	seasonal_prcp	seasonal_tmax	seasonal_tmin
86330	1981	1	458.9	30.32778	19.151111
86330	1981	2	370.4	26.44674	14.951087
86330	1981	3	211.2	19.76630	9.120652
86330	1981	4	272.0	24.73516	13.093407
86330	1982	1	465.9	30.33667	17.764444
86330	1982	2	229.0	26.74130	14.385870

- 826 En la siguiente Figura se muestra la variación de la precipitación acumulada estival para las
 827 15 estaciones utilizadas.

```

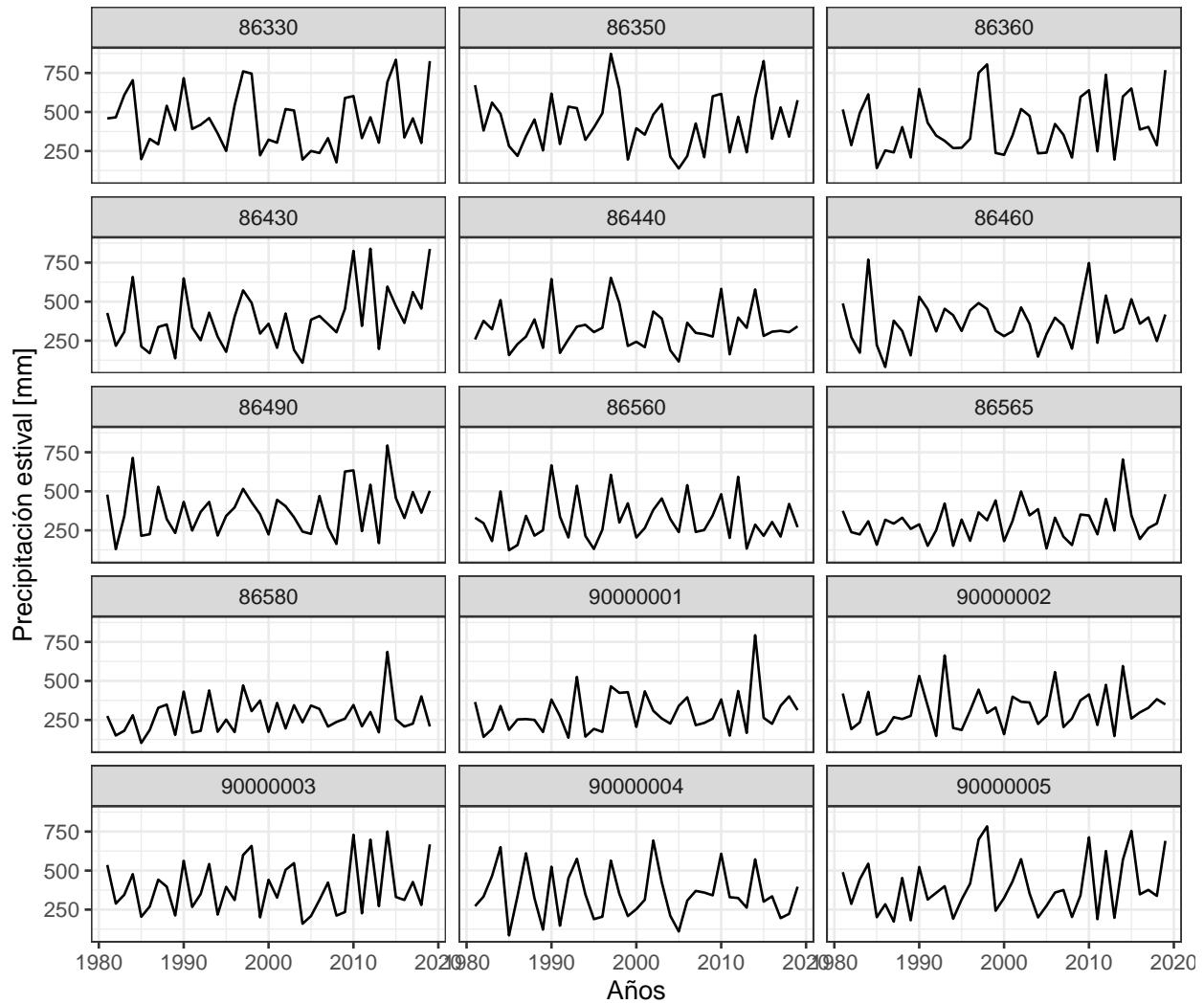
ggplot2::ggplot(data = seasonal_covariates %>%
  dplyr::filter(season == 1),
  ggplot2::aes(x = year, y = seasonal_prcp)) +

```

```

ggplot2::geom_line() +
ggplot2::facet_wrap(~station_id, ncol = 3) +
ggplot2::theme_bw() +
ggplot2::labs(x = 'Años', y = 'Precipitación estival [mm]')

```



828

829 Se observa en la figura que si bien hay cierta coincidencia, no todos los picos y valles ocurren
 830 en el mismo momento. Utilizar un método que genere el tiempo local considerando estas
 831 variaciones espaciales es muy útil para representar fehacientemente los patrones espaciales y
 832 temporales.

Al correr la función se realiza el ajuste de los cuatro modelos

```

# para cada una de las estaciones. En este caso, por cuestiones de tiempo a cargar un
# Si el usuario desea correrlo deberá ver la nota anterior.

gamgen_fit <- gamwgen::local_calibrate(
  climate = climate,
  # Registro histórico de variables meteorológicas
  stations = stations,
  # Estaciones meteorológicas
  seasonal_covariates = seasonal_covariates,
  # Totales trimestrales de precipitación
  control = control_fit,
  # Objeto de control
  verbose = FALSE)
  # Impresión de mensajes en la consola.

```

- 833 Para esta demostración, cargamos el objeto con el ajuste del modelo ya realizado.

```

# Copiamos el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('input_data/local/fit_local_correlated_weather.RData')) {
  fs::file_copy(system.file('/autorun/local', "fit_local_correlated_weather.RData",
    package = "gamwgen"),
    new_path = 'input_data/local/fit_local_correlated_weather.Rdata')
}

# Cargamos el archivo recientemente creado
load('input_data/local/fit_local_correlated_weather.RData')

# Clase del objeto con el ajuste del generador
class(gamgen_fit)

```

```

834 ## [1] "gamwgen"

# Contenido del modelo

names(gamgen_fit)

835 ## [1] "control"           "stations"          "climate"
836 ## [4] "seasonal_covariates" "crs_used_to_fit"   "start_climatology"
837 ## [7] "fitted_models"       "models_data"        "models_residuals"
838 ## [10] "statistics_threshold" "exec_times"

```

839 Dentro del objeto se guardan todo lo necesario para la simulación así como información
 840 accesoría.

- 841 • **control**: copia de la configuración usada para calibrar el generador
- 842 • **stations**: estaciones meteorológicas utilizadas para la calibración
- 843 • **climate**: datos climáticos de cada uno de las estaciones
- 844 • **seasonal_covariates**: series temporales de totales trimestrales de precipitación y
 845 medias trimestrales de temperaturas máxima y mínima.
- 846 • **crs_used_to_fit**: sistema de referencia espacial usado para proyectar
- 847 • **start_climatology**: climatología diaria de cada una de las variables de entrada.
- 848 • **fitted_models**: modelos ajustados, uno para cada variable: temperaturas máxima y
 849 mínima y ocurrencia y montos de precipitación.
- 850 • **models_data**: datos usados efectivamente usados para ajustar los modelos (sin NAs)
- 851 • **models_residuals**: residuos de cada uno de los modelos. Es decir, la diferencia entre
 852 el valor ajustado por el modelo (clima local) y el valor observado en el día **i**
- 853 • **statistics_threshold**: umbrales de amplitud térmica diaria por mes. Si la amplitud
 854 simulada está fuera de este rango, se repetirá la simulación para ese día a los fines de
 855 mantener la consistencia entre variables

856 • `exec_times`: tiempo de ejecución de cada una de las etapas del ajuste Al inspeccionar
857 el objeto con los resultados se puede ver que cada estación meteorológica tiene su
858 propia sublistas donde se almacenan los modelos para cada una de ellas.

```
names(gamgen_fit$fitted_models)
```

```
859 ## [1] "86330"     "86350"     "86360"     "86430"     "86440"     "86460"  
860 ## [7] "86490"     "86560"     "86565"     "86580"     "90000001" "90000002"  
861 ## [13] "90000003" "90000004" "90000005"
```

862 Cada uno de los GAMs ajustados se almacenan en el objeto `gamgen_fit` y pueden ser
863 evaluados con la función `summary()`.

```
summary(gamgen_fit$fitted_models$`86330`$tmax_fit)
```

```
864 ##  
865 ## Family: gaussian  
866 ## Link function: identity  
867 ##  
868 ## Formula:  
869 ## tmax ~ s(tmax_prev, tmin_prev, k = 50) + s(prcp_occ, bs = "re") +  
870 ##      s(prcp_occ_prev, bs = "re") + s(doy, bs = "cc", k = 30) +  
871 ##      s(SX1, SN1, k = 20) + s(SX2, SN2, k = 20) + s(SX3, SN3, k = 20) +  
872 ##      s(SX4, SN4, k = 20)  
873 ##  
874 ## Parametric coefficients:  
875 ##                      Estimate Std. Error t value Pr(>|t|)  
876 ## (Intercept) 26.18758    0.03397   770.9 <2e-16 ***  
877 ## ---
```

```

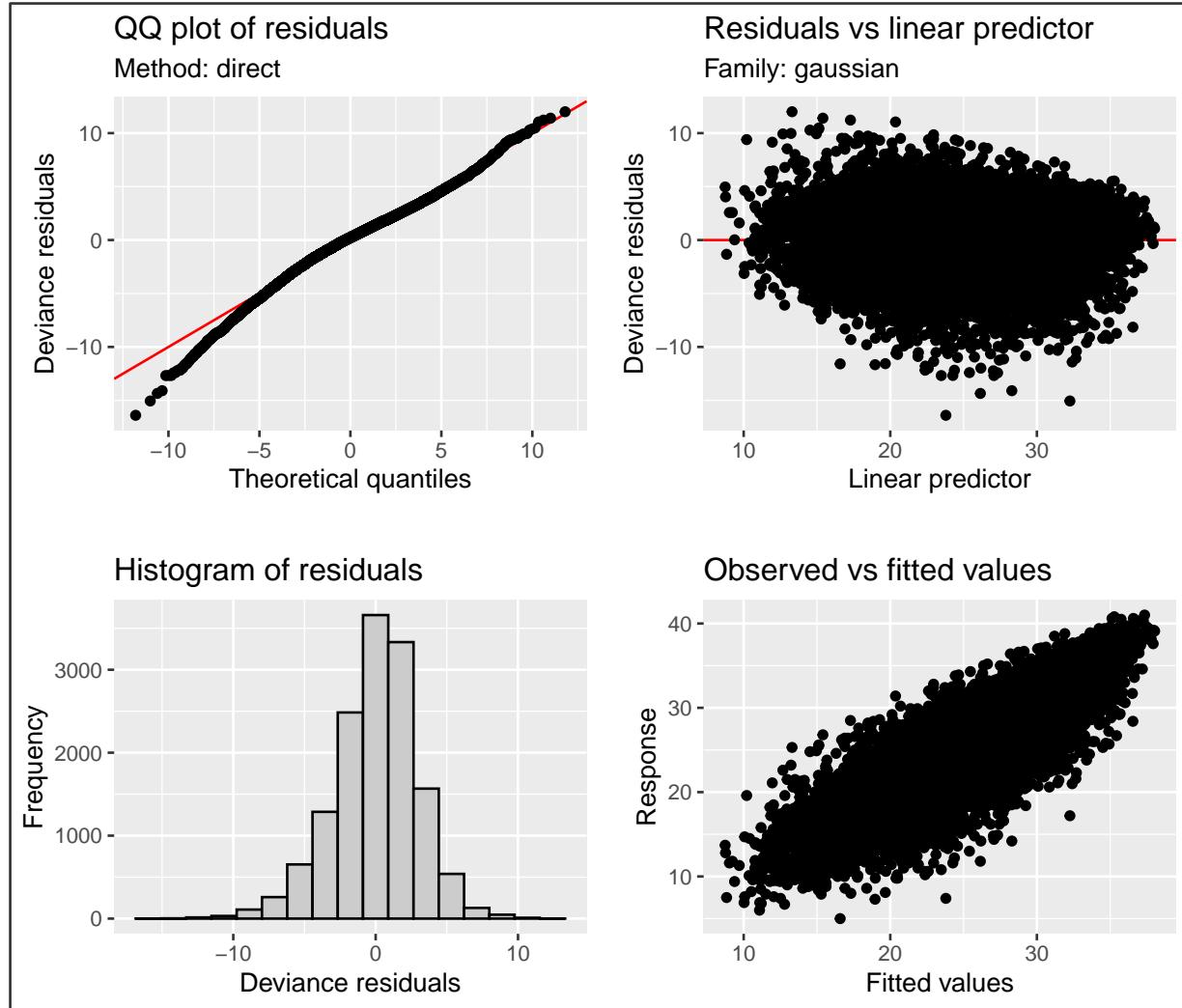
878 ## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
879 ##
880 ## Approximate significance of smooth terms:
881 ##          edf Ref.df      F p-value
882 ## s(tmax_prev,tmin_prev) 32.4286 40.704 235.57 <2e-16 ***
883 ## s(prcp_occ)           0.9979  1.000 520.81 <2e-16 ***
884 ## s(prcp_occ_prev)     0.9987  1.000 866.16 <2e-16 ***
885 ## s(doy)                10.3038 28.000  33.61 <2e-16 ***
886 ## s(SX1,SN1)            3.6312  4.458  26.92 <2e-16 ***
887 ## s(SX2,SN2)            2.0001  2.000  36.90 <2e-16 ***
888 ## s(SX3,SN3)            4.3491  5.631  21.09 <2e-16 ***
889 ## s(SX4,SN4)            2.0007  2.001  40.04 <2e-16 ***
890 ## ---
891 ## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
892 ##
893 ## R-sq.(adj) = 0.778 Deviance explained = 77.9%
894 ## fREML = 35496 Scale est. = 8.8219 n = 14128

```

895 La función **summary** permite analizar los resultados del ajuste de cada uno de los modelos.
896 Para el caso del modelo de temperatura máxima podemos ver la fórmula del GAM en la
897 parte superior bajo el apartado **Formula** y la significancia de cada uno de los términos
898 del modelo en la tabla inmediatamente inferior. En este configuración, al incluirle variables
899 estacionales, se incorporan nuevos términos el modelo como SX1, SX2, SX3, SX4 y SN1, SN2,
900 SN3, SN4, que corresponden a variables dummy de las medias trimestrales de temperatura
901 máxima y mínima, respectivamente. Se puede observar que todos los términos son altamente
902 significativos. También se incluyen como pruebas de bondad del ajuste el porcentaje de la
903 varianza explicada por el modelo y el valor de R-ajustado.

904 El paquete `gratia` (Simpson, Gavin (2018)) es muy útil para la visualización de los ajustes
905 de manera gráfica. Esta función toma los diagnósticos por defecto de la función `gam.check`
906 del paquete `mgcv` (Wood, Simon (2001)) y los convierte en gráficos de la librería `ggplot2`
907 (Wickham, Hadley(2011)) que son visualmente muy atractivos. La figura está dividida en
908 cuatro paneles, cada uno mostrando un diagnóstico diferente. En el panel superior izquierdo
909 se muestra un Q-Q Plot de los residuos. Si el modelo ha ajustado satisfactoriamente los
910 datos, los puntos deberían estar sobre la recta 1:1 demarcada en rojo. El panel superior
911 derecho muestra los residuos vs el predictor lineal. El objetivo de este diagnóstico es que los
912 residuos se distribuyan al azar y que no tengan un patrón claro. La presencia de patrones
913 en los residuos indicaría que el modelo no ha explicado algún componente importante de
914 la variabilidad de los datos. En el panel inferior izquierdo se muestra un histograma de los
915 residuos siendo deseable que lo mismos tengan una distribución próxima a la Normal. El
916 último gráfico en el panel inferior derecho muestra una gráfica de dispersión entre los valores
917 ajustados y observados.

```
gratia::appraise(gamgen_fit$fitted_models`^86330`$tmax_fit) +  
  ggplot2::theme_bw()
```



918

919 Estos diagnósticos exploratorios pueden aplicarse a cada uno de los modelos ajustados por la
920 función `local_calibrate`.

921 Con la creación del objeto `gamgen_fit` finaliza el proceso de ajuste y ya se pueden generar
922 series sintéticas para la o las estaciones usadas para calibrar el generador.

923 **5.3.4.3 Generación de series** La generación de series sigue la misma estructura ante-
924rior, una función para configurar la generación y otra que realiza la generación propiamente
925 dicha.

926 Los argumentos de la función de control son:

- **nsim**: cantidad de simulaciones a realizar. Se debe ingresar un valor **entero** mayor o igual a 1.
- **seed**: semilla. Se debe ingresar cualquier numero **entero**. No es necesario recordarlo porque se guarda junto a los resultados.
- **avbl_cores**: cantidad de núcleos disponibles para la paralelización.
- **use_spatially_correlated_noise**: utilizar la generación estocástica espacialmente correlacionada. Esta opción sólo es válida si en el ajuste y en la simulación se usaron más de cinco estaciones meteorológicas diferentes. Con un menor número no es posible calcular los variogramas necesarios para la generación de los campos aleatorios. Se debe introducir un **boolean** (TRUE or FALSE).
- **use_temporary_files_to_save_ram**: si se simulan muchas realizaciones o los recursos informáticos son escasos, esta opción permite guardar los resultados de cada una de las realizaciones en el disco liberando memoria RAM que quedará disponible para generar nuevas simulaciones. Al finalizar la generación todos los archivos se combinan en uno único. Se debe introducir un **boolean** (TRUE or FALSE).
- **use_temporary_files_to_save_ram**: esta opción permite eliminar los archivos temporales creados para ahorrar RAM luego de terminar la generación de todas las simulaciones. Se debe introducir un **boolean** (TRUE or FALSE).

```

control_sim <- gamwgen::local_simulation_control(
  nsim = 10,
  # Cantidad de simulaciones a realizar
  seed = 1234,
  # Semilla para que los resultados sean reproducibles
  avbl_cores = 6,
  # Cantidad de núcleos disponibles a utilizar
  use_spatially_correlated_noise = TRUE,
  # Usar modelo de ruido espacialmente correlacionado
)

```

```

use_temporary_files_to_save_ram = TRUE,
# Guardar resultados intermedios para ahorrar RAM
remove_temp_files_used_to_save_ram = TRUE)

# Borrar los resultados intermedios creados anteriormente

```

945 Luego se procede a la simulación de datos meteorológicos. Los argumentos de la función de
 946 simulación son:

- 947 • **model**: objeto con el resultado de la función `local_calibrate()`
- 948 • **simulation_locations**: objeto tipo `sf` con la ubicación de las estaciones a simular. Las estaciones usadas deben haber sido incluidas en el proceso de ajuste. No es necesario que todas estén presentes, se pueden generar series solo sobre algunas de ellas.
- 952 • **start_date**: fecha de comienzo de la generación de series sintéticas. Si no se incluyeron covariables estacionales en el ajuste, la fecha de comienzo es completamente arbitraria. Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables, ni tampoco posterior. Se debe introducir una fecha en formato `date`
- 956 • **end_date**: fecha de fin de la generación de series sintéticas. Si no se incluyeron covariables estacionales en el ajuste, la fecha de fin es completamente arbitraria. Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables, tampoco puede ser posterior. Se debe introducir una fecha en formato `date`
- 960 • **control**: objeto de control creado con la función `control_sim()`.
- 961 • **output_folder**: ruta al directorio donde se guardarán los resultados, tanto finales como intermedios.
- 963 • **output_filename**: nombre del archivo de salida. Para facilitar la interoperabilidad, el archivo generado es un archivo de texto en formato separado por comas (.csv)
- 965 • **seasonal_covariates**: datos agregados trimestrales. Si el ajuste se realizó con covariables, la generación también debe realizarse con ellas. Caso contrario se producirá

967 un error. Se debe introducir un data frame con los valores agregados para las tres
968 variables (precipitación y temperaturas máxima y mínima) pero no necesariamente
969 deben ser los mismos a los utilizados en el ajuste. Si se desean simular tendencias de
970 algún tipo, ya sea de un modelo de cambio climático o arbitrarias, se deben perturbar
971 estas variables trimestrales e introducirlas aquí. Estas series si deben tener la misma
972 longitud que el período a generar

- 973 • **verbose**: controla la impresión de mensajes en la consola. FALSE por defecto.

```
# Al correr la función se realiza la generación de series para
# cada una de las estaciones.

# En este caso, por cuestiones de tiempo, vamos a cargar un
# objeto con los resultados de la simulación

simulated_climate <- gamwgen::local_simulation(
  model = gamgen_fit,
  # Objeto con los resultados del ajuste
  simulation_locations = stations,
  # Estaciones para las cuales simular
  start_date = as.Date('2019-01-01'),
  # Fecha de comienzo de las simulaciones
  end_date = as.Date('2019-12-31'),
  # Fecha de fin de las simulaciones
  control = control_sim,
  # Objeto con la configuración
  output_folder = getwd(),
  # Directorio donde se guardarán los resultados
  output_filename = 'simulations.csv',
  # Nombre del archivo de salida
  seasonal_covariates = seasonal_covariates,
```

```

# Covariables estacionales
verbose = FALSE)

# Impresión de mensajes en la consola

```

974 Esta función produce dos tipos de resultados: una lista que permanece en el ambiente de R y
 975 los datos generados que son guardados como .csv en el directorio indicado precedentemente.

```

# Copiamos el archivo preajustado a nuestro directorio de trabajo

if (!fs::file_exists('output_data/local/simulated_local_correlated_weather.RData')) {
  fs::file_copy(system.file('/autorun/local', "simulated_local_correlated_weather.RData",
                           new_path = 'output_data/local/simulated_local_correlated_weather.RData')
}

# Cargamos el archivo recientemente creado

load('output_data/local/simulated_local_unconditional.RData')

# Clase del objeto con el ajuste del generador

class(simulated_climate)

976 ## [1] "list"           "gamwgen.climate"

# Contenido del modelo

names(simulated_climate)

977 ## [1] "nsim"
978 ## [2] "seed"
979 ## [3] "realizations_seeds"
980 ## [4] "simulation_points"
981 ## [5] "output_file_with_results"
982 ## [6] "output_file_fomart"

```

```
983 ## [7] "rdata_file_with_fitted_stations_and_climate"  
984 ## [8] "exec_times"
```

985 La lista contiene los siguientes objetos:

- 986 • `nsim`: cantidad de realizaciones.
- 987 • `seed`: semilla general para toda la generación. Corresponde a la que se incluye en la
988 función de control.
- 989 • `realization_seeds`: semillas para cada una de las realizaciones. Esto permite replicar
990 los resultados.
- 991 • `simulation_points`: puntos donde se generaron las series sintéticas.
- 992 • `output_file_with_results`: nombre del archivo con los resultados.
- 993 • `output_file_format`: tipo de archivo de salida, en este caso `.csv`.
- 994 • `rdata_file_with_fitted_stations_and_climate`: archivo `.RData` con los datos me-
995 teorológicos observados que fueron utilizados en el ajuste. También se incluyen los
996 metadatos de cada uno de esos puntos.
- 997 • `exec_times`: tiempo de ejecución de la generación.

998 Ahora veremos el formato del archivo de salida que contiene las series sintéticas.

999 Para desmenuzar la generación del tiempo local se pueden correr algunas de las funciones
1000 que forman parte de `local_simulation`. Por ejemplo, del objeto `gamgen_fit` se extraen
1001 los residuos y con la función `gamwgen:::generate_residuals_statistics` se calculan los
1002 parámetros.

1003 En la tabla anterior se muestran los parámetros necesarios para generar el tiempo local de
1004 las temperaturas máxima y mínima.

```
gen_noise_params <- gamwgen:::generate_month_params(  
  residuals = gamgen_fit$models_residuals,
```

```

observed_climate = gamgen_fit$models_data,
stations = gamgen_fit$stations)

# Ejemplo de variograma de residuos de temperatura máxima para días secos

gen_noise_params[[1]]$variogram_parameters$tmax_dry

1005 ## [1] 0.000000 6.978337 495.779099

```

1006 Para cada una de las variables, precipitación y temperaturas máxima y mínima, se ajusta
 1007 un variograma por máxima verosimilitud que permite representar la variabilidad espacial
 1008 de los residuos de los modelos GAMs, que corresponden al tiempo local. En el variograma
 1009 se muestran los tres parámetros **nugget**, **psill** y **range**. El rango de los datos es de 500
 1010 km, lo que es lógico si se considera que los puntos están distribuidos homogeneamente en la
 1011 República Oriental del Uruguay.

1012 En la siguiente FIgura se muestra la variabilidad espacial del tiempo local de temperatura
 1013 del día 1 de enero de 2019 para días secos. Cabe mencionar que se crean también los mismos
 1014 datos para los días húmedos y en función de la ocurrencia de precipitación se selecciona uno
 1015 u otro.

```
RandomFields::RFoptions(printlevel = 0, spConform = FALSE)
```

```

temp_local <- control_sim$temperature_noise_generating_function(simulation_points =
  dplyr::mutate(
    gen_noise_params
    month_number = 1L
    selector = 'Dry',
    seed = NULL) %>%

```

```

dplyr::mutate(date = as.Date(paste0('2019-', '01', '-', '01')))) %>%
tidyrr::gather(variable, valor, -c('date', 'geometry'))

# Mapa de Uruguay

uruguay <- raster::getData('GADM', country='URY', level=1) %>%
sf::st_as_sf(.) %>%
sf::st_transform(crs = 32721)

# Paleta de colores

colr <- grDevices::colorRampPalette(RColorBrewer::brewer.pal(9, 'YlOrRd'))

# Quiebres de la escala de colores

quiebres <- c(-8.0, -6.0, -4.0,-2.0, 0, 2.0, 4.0, 6.0, 8.0)

# Etiquetas para los colores

etiquetas <- c('-8', '-6', '-4', '-2', '0', '2', '4', '6', '8')

# Etiquetas de las facetas

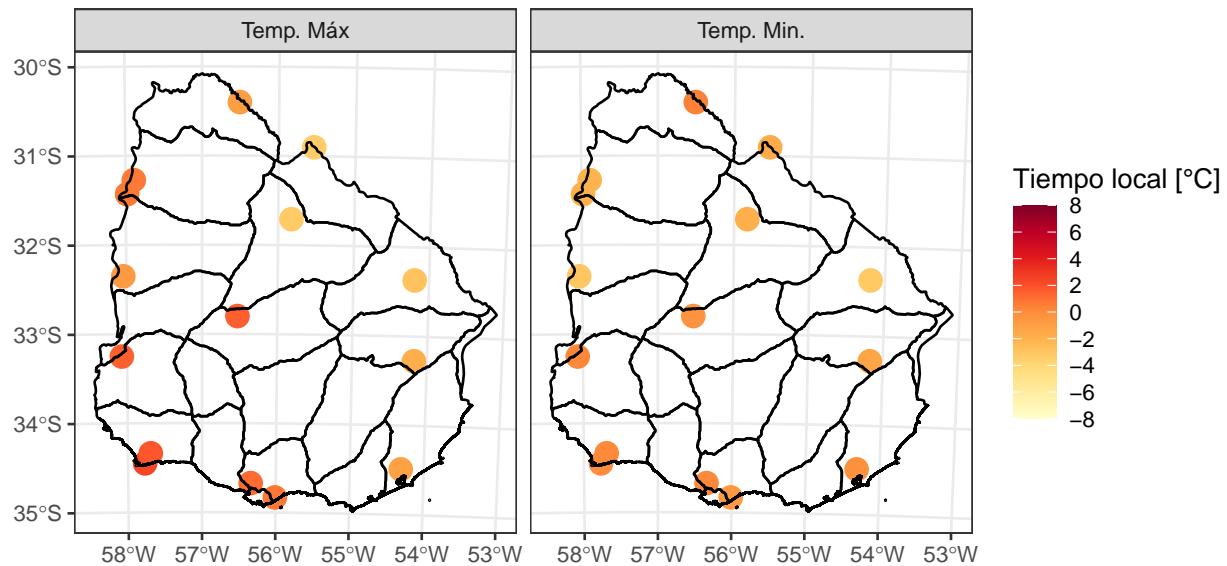
labs <- c("Temp. Máx", "Temp. Min.")

names(labs) <- c("tmax_residuals", "tmin_residuals")

ggplot2::ggplot(data = temp_local) +
ggplot2::geom_sf(ggplot2::aes(color = valor), size = 4) +
ggplot2::scale_color_gradientn(name = "Tiempo local [°C]",
                               colours = colr(9),
                               breaks = quiebres,
                               labels = etiquetas,
                               limits = c(-8, 8)) +
ggplot2::geom_sf(data = uruguay, fill = NA, color = "black", inherit.aes = FALSE) +

```

```
ggplot2::facet_wrap(~variable, labeller = ggplot2::labeller(variable = labs)) +
  ggplot2::theme_bw()
```



1016

1017 Se puede observar como el tiempo local tiene una estructura espacial que es representada a
 1018 través del variograma y permite que los resultados para las distintas estaciones sean espa-
 1019 cialmente consistentes. Esto quiere decir que estaciones meteorológicas cercanas tenderán a
 1020 tener valores de tiempo local similares.

1021 **5.4 Generación de series sintéticas para una grilla regular**

1022 La generación de series sobre grillas regular es muy similar a lo mostrado anteriormente.
1023 Los fundamentos son los mismos, sólo los modelos estadísticos que modelan el clima local
1024 incorporan un nuevo término que permite incluir la dimensión espacial.

1025 **5.4.0.1 Crear archivos de entrada** Para este ejemplo se utilizan datos de varias esta-
1026 ciones pero el formato de los mismos es igual a los mostrados anteriormente.

1027 Los archivos necesarios son:

- 1028 • stations.csv
1029 • climate.csv

1030 Los datos meteorológicos se dividen en dos archivos separados: `stations.csv` y
1031 `climate.csv`. Los nombres de los mismos no deben ser necesariamente iguales a los
1032 usados aquí.

1033 Los metadatos de las estaciones se alojan en el archivo `stations.csv`. Este archivo contiene
1034 la información de las estaciones meteorológicas que serán usadas en el ajuste del modelo.
1035 Las variables que deben ser incluidas en la tabla son:

- 1036 • station_id: número único para cada estación meteorológica. La variable debe ser
1037 de tipo *integer*
1038 • latitude: latitud en grados decimales. La variable debe ser de tipo *double*
1039 • longitude: longitud en grados decimales. La variable debe ser de tipo *double*

1040 La tabla puede tener más variables pero sólo se necesitan las anteriores.

1041 A continuación se muestran la primera fila del dataset y los tipo de datos de cada una de
1042 las variables.

```
# Vista de los metadatos de la estación
knitr::kable(head(stations), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center")
```

station_id	nombre	lat_dec	lon_dec	elev
86330	Artigas	-30.40	-56.51	120.38
86350	Rivera	-30.90	-55.54	241.94
86360	Salto	-31.43	-57.98	41.00
86430	Paysandú	-32.35	-58.04	61.12
86440	Melo	-32.37	-54.19	100.36
86460	Paso de los Toros	-32.80	-56.53	75.48

1043 El objeto **stations** debe ser convertido de *tibble* a *sf*. El sistema de referencia espacial debe
1044 ser planar. No es necesario un sistema de referencia espacial en particular, solamente las
1045 coordenadas deben estar expresadas en metros.

```
# Convertimos el objeto stations a sf y se convierte su proyección de WGS 1984 a
# UTM Zona 21 S
stations %<>%
  sf::st_as_sf(coords = c("lon_dec", "lat_dec"), crs = 4326) %>%
  sf::st_transform(crs = 32721)
```

1046 En el siguiente mapa muestra la distribución de las estaciones meteorológicas usadas en el
1047 ejemplo.

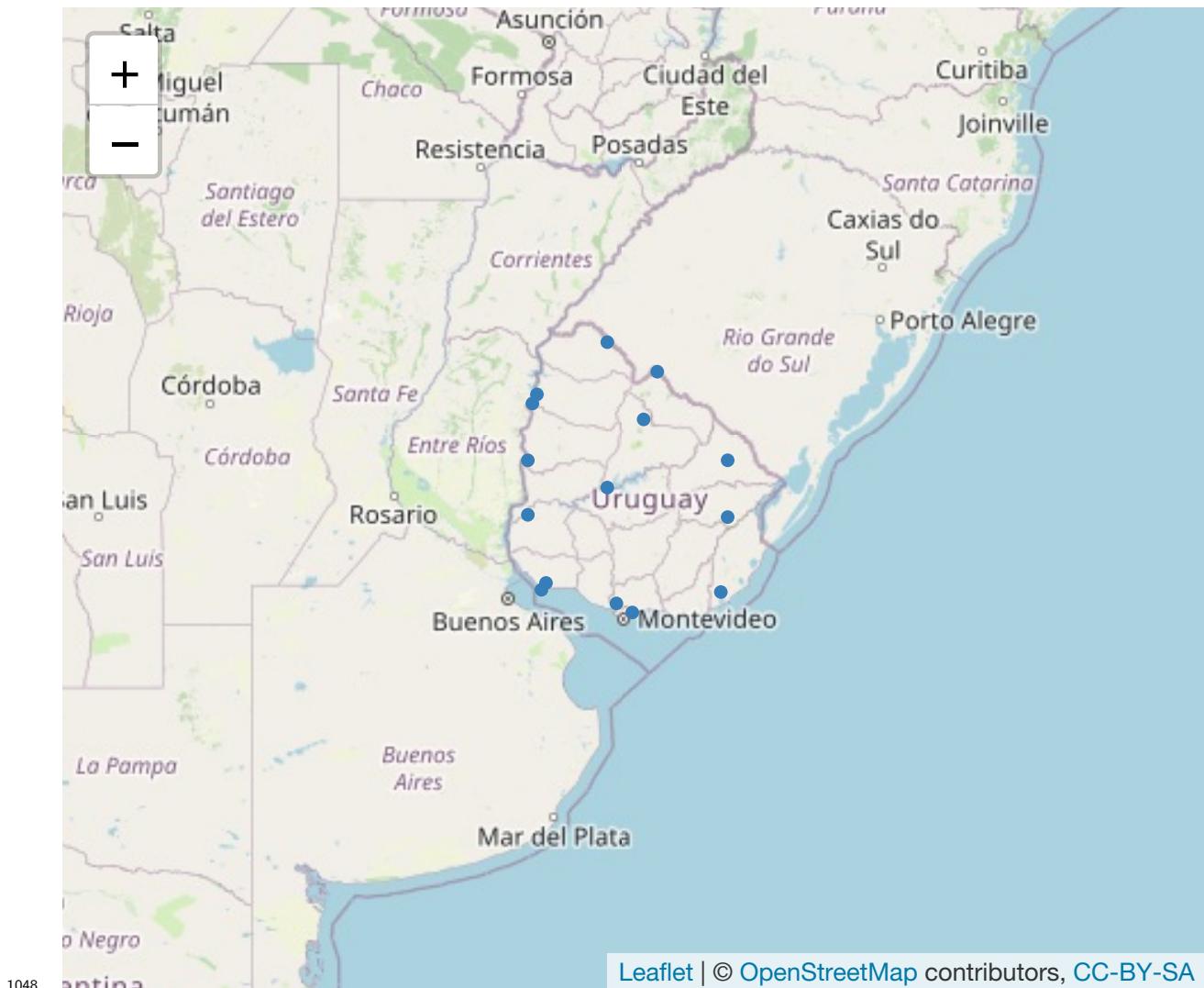
```
# Convertir el objeto con las estaciones a coordenadas geográficas
stations_geo <- stations %>%
```

```

sf::st_transform(crs = 4326) %>%
dplyr::mutate(lat = sf::st_coordinates(.)[,2],
             lon = sf::st_coordinates(.)[,1])

# Creación del mapa con la distribución de las estaciones
leaflet::leaflet(data = stations_geo) %>%
  leaflet::addTiles() %>%
  leaflet::setView(lat = -33, lng = -56, zoom = 5) %>%
  leaflet::addCircleMarkers(lat = ~lat, lng = ~lon, radius = 3,
                           fillColor = "#377eb8",
                           color = "#377eb8",
                           stroke = FALSE, fillOpacity = 1, opacity = 1,
                           popup = ~sprintf("<b>%s (%d)</b><br>Lat.: %.3f<br>Lon.: %.
                           nombre, station_id, lat, lon, elev))

```



1049 La información climática se aloja en el archivo `climate.csv`. Este archivo contiene los datos
 1050 de las estaciones meteorológicas que serán usadas en el ajuste del modelo. Las variables que
 1051 deben ser incluidas en la tabla son:

- 1052 • `date`: fecha del dato. La variable debe ser de tipo `date`
- 1053 • `station_id`: número único para cada estación meteorológica. La variable debe ser
 de tipo `integer`
- 1055 • `prcp`: datos diarios de precipitación La variable debe ser de tipo `double`
- 1056 • `tmax`: datos diarios de temperatura máxima. La variable debe ser de tipo `double`
- 1057 • `tmin`: datos diarios de temperatura mínima. La variable debe ser de tipo `double`

1058 A continuación se muestran las primeras cinco filas del dataset y los tipos de datos de cada
1059 una de las variables.

```
knitr::kable(head(climate), "latex", booktabs = T) %>%  
  kableExtra::kable_styling(position = "center")
```

date	station_id	tmax	tmin	prcp
1981-01-01	86330	34.6	21.7	0.0
1981-01-02	86330	33.8	21.2	19.5
1981-01-03	86330	28.3	19.4	0.0
1981-01-04	86330	30.3	17.4	0.0
1981-01-05	86330	33.4	17.4	21.0
1981-01-06	86330	32.8	20.4	8.0

1060 En total se utilizan 15 estaciones meteorológicas, 10 provistas por INUMET (Instituyo
1061 Uruguayo de Meteorología) y 5 por INIA (Instituto Nacional de Investigación Agropecuaria).

```
unique(climate$station_id)  
  
1062 ## [1] 86330 86350 86360 86430 86440 86460 86490 86560  
1063 ## [9] 86565 86580 90000001 90000002 90000003 90000004 90000005
```

1064 **5.4.0.2 Ajuste de los modelos** El ajuste de los modelos es igual que para los dos
1065 ejemplos antes mostrados. Se utiliza la función `control_fit` para la configuración del ajuste
1066 y `spatial_calibrate` para realizar el ajuste.

```
control_fit <- gamwgen::spatial_fit_control(  
  prcp_occurrence_threshold = 0.1, # Umbral para la definición de días húmedos
```

```

avbl_cores = 6, # Cantidad de núcleos disponibles
planar_crs_in_metric_coords = 32721) # Sistema de referencia espacial (en metros)

1067 Este ejemplo se realizará utilizando covriables trimestrales pero es válido para series esta-
1068 cionarias.

```

Agregación de valores diarios

```

seasonal_covariates <- gamwgen::summarise_seasonal_climate(climate, umbral_faltantes =
                                                               ...)

# Se muestran las primeras cinco filas
knitr::kable(head(seasonal_covariates), "latex", booktabs = T) %>%
  kableExtra::kable_styling(position = "center")

```

station_id	year	season	seasonal_prcp	seasonal_tmax	seasonal_tmin
86330	1981	1	458.9	30.32778	19.151111
86330	1981	2	370.4	26.44674	14.951087
86330	1981	3	211.2	19.76630	9.120652
86330	1981	4	272.0	24.73516	13.093407
86330	1982	1	465.9	30.33667	17.764444
86330	1982	2	229.0	26.74130	14.385870

```

# Al correr la función se realiza el ajuste de los cuatro modelos para cada una de
# las estaciones. En este caso, por cuestiones de tiempo a cargar un objeto ya precalc-
# Si el usuario desea correrlo deberá ver la nota anterior.

```

```

gamgen_fit <- gamwgen::spatial_calibrate(climate = climate, # Registro histórico de var-
                                             stations = stations, # Estaciones meteorológicas
                                             seasonal_covariates = seasonal_covariates, # Totales trimestrales de precipitación

```

```

control = control_fit, # Objeto de control
verbose = FALSE) # Impresión de mensajes en la consola.

1069 Para esta demostración, cargamos el objeto con el ajuste del modelo ya realizado.

# Copiamos el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('input_data/spatial/fit_spatial_conditional.RData')) {
  fs::file_copy(system.file('/autorun/spatial', "fit_spatial_conditional.RData", package =
    new_path = 'input_data/spatial/fit_spatial_conditional.Rdata')
}

# Cargamos el archivo recientemente creado
load('input_data/spatial/fit_spatial_conditional.RData')

# Clase del objeto con el ajuste del generador
class(gamgen_fit)

1070 ## [1] "gamwgen"

# Contenido del modelo
names(gamgen_fit)

1071 ## [1] "control"           "stations"          "climate"
1072 ## [4] "seasonal_covariates" "crs_used_to_fit"   "start_climatology"
1073 ## [7] "fitted_models"       "models_data"       "models_residuals"
1074 ## [10] "statistics_threshold" "exec_times"

```

1075 Dentro del objeto se guardan todo lo necesario para la simulación así como información
 1076 accesoria.

- **control**: copia de la configuración usada para calibrar el generador
- **stations**: estaciones meteorológicas utilizadas para la calibración
- **climate**: datos climáticos de cada uno de las estaciones
- **seasonal_covariates**: series temporales de totales trimestrales de precipitación y medias trimestrales de temperaturas máxima y mínima.
- **crs_used_to_fit**: sistema de referencia espacial usado para proyectar
- **start_climatology**: climatología diaria de cada una de las variables de entrada.
- **fitted_models**: modelos ajustados, uno para cada variable: temperaturas máxima y mínima y ocurrencia y montos de precipitación.
- **models_data**: datos usados efectivamente usados para ajustar los modelos (sin NAs)
- **models_residuals**: residuos de cada uno de los modelos. Es decir, la diferencia entre el valor ajustado por el modelo (clima local) y el valor observado en el día **i**
- **statistics_threshold**: umbrales de amplitud térmica diaria por mes. Si la amplitud simulada está fuera de este rango, se repetirá la simulación para ese día a los fines de mantener la consistencia entre variables
- **exec_times**: tiempo de ejecución de cada una de las etapas del ajuste

A diferencia de la configuración que ajusta distintos modelos para cada estación meteorológica, en este caso se ajusta un sólo modelo para toda la región de interés. Al visualizar lo que está almacenado en la sublista **fitted_models** se ve que solo hay cuatro GAMs, dos para la precipitación y dos para las temperaturas máxima y mínima.

```
names(gamgen_fit$fitted_models)
```

```
1097 ## [1] "prcp_occ_fit" "prcp_amt_fit" "tmax_fit"      "tmin_fit"
```

Cada uno de los GAMs ajustados se almacenan en el objeto **gamgen_fit** y pueden ser evaluados con la función **summary()**.

```

summary(gamgen_fit$fitted_models$tmax_fit)

1100 ## 
1101 ## Family: gaussian
1102 ## Link function: identity
1103 ##
1104 ## Formula:
1105 ## tmax ~ te(tmax_prev, tmin_prev, longitude, latitude, d = c(2,
1106 ##      2), k = length(unique_stations)) + te(type_day, longitude,
1107 ##      latitude, d = c(1, 2), bs = c("re", "tp"), k = length(unique_stations)) +
1108 ##      te(type_day_prev, longitude, latitude, d = c(1, 2), bs = c("re",
1109 ##      "tp"), k = length(unique_stations)) + te(doy, longitude,
1110 ##      latitude, d = c(1, 2), bs = c("cc", "tp"), k = length(unique_stations)) +
1111 ##      te(SX1, SN1, longitude, latitude, d = c(2, 2), k = length(unique_stations)) +
1112 ##      te(SX2, SN2, longitude, latitude, d = c(2, 2), k = length(unique_stations)) +
1113 ##      te(SX3, SN3, longitude, latitude, d = c(2, 2), k = length(unique_stations)) +
1114 ##      te(SX4, SN4, longitude, latitude, d = c(2, 2), k = length(unique_stations))
1115 ##
1116 ## Parametric coefficients:
1117 ##             Estimate Std. Error t value Pr(>|t|)
1118 ## (Intercept) 25.91     29.52   0.878    0.38
1119 ##
1120 ## Approximate significance of smooth terms:
1121 ##                                         edf Ref.df      F p-value
1122 ## te(tmax_prev,tmin_prev,longitude,latitude) 76.902 86.484 1267.40 <2e-16 ***
1123 ## te(type_day,longitude,latitude)            27.680 29.000 161.22 <2e-16 ***
1124 ## te(type_day_prev,longitude,latitude)       14.703 15.000  726.24 <2e-16 ***
1125 ## te(doy,longitude,latitude)                47.907 195.000   77.44 <2e-16 ***

```

```

1126 ## te(SX1,SN1,longitude,latitude)      18.355 23.456 75.14 <2e-16 ***
1127 ## te(SX2,SN2,longitude,latitude)      6.005  6.009 184.32 <2e-16 ***
1128 ## te(SX3,SN3,longitude,latitude)      30.642 36.395 48.54 <2e-16 ***
1129 ## te(SX4,SN4,longitude,latitude)      6.002  6.004 193.99 <2e-16 ***
1130 ## ---
1131 ## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
1132 ##
1133 ## R-sq.(adj) = 0.779 Deviance explained = 77.9%
1134 ## fREML = 5.221e+05 Scale est. = 8.9846 n = 207269

```

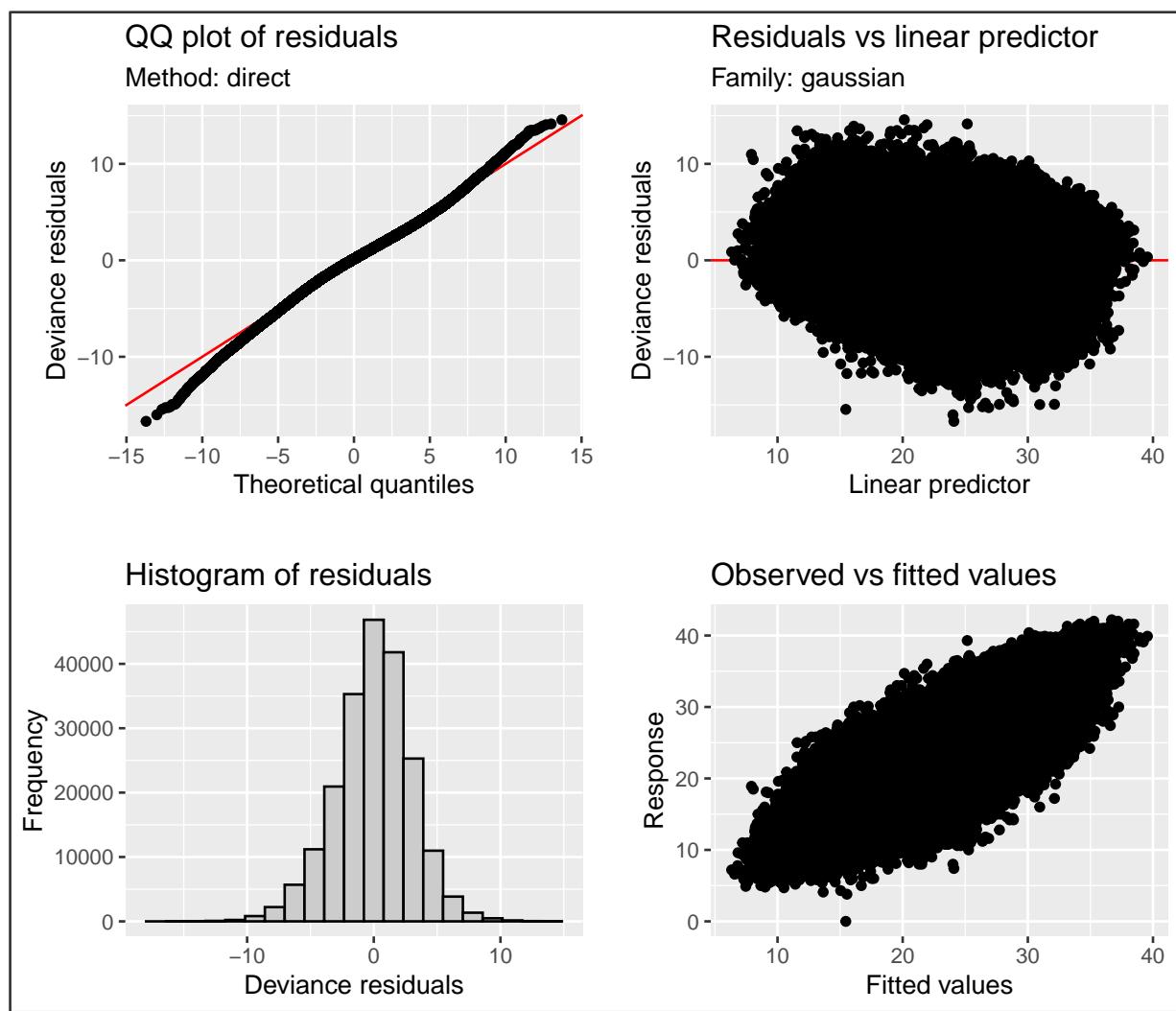
1135 La función **summary** permite analizar los resultados del ajuste de cada uno de los modelos.
 1136 Para el caso del modelo de temperatura máxima podemos ver la fórmula del GAM en la
 1137 parte superior bajo el apartado **Formula** y la significancia de cada uno de los términos
 1138 del modelo en la tabla inmediatamente inferior. En este configuración, al incluirle variables
 1139 estacionales, se incorporan nuevos términos el modelo como SX1, SX2, SX3, SX4 y SN1, SN2,
 1140 SN3, SN4, que corresponden a variables dummy de las medias trimestrales de temperatura
 1141 máxima y mínima, respectivamente. Se puede observar que todos los términos son altamente
 1142 significativos. También se incluyen como pruebas de bondad del ajuste el porcentaje de la
 1143 varianza explicada por el modelo y el valor de R-ajustado.

1144 El paquete **gratia** (Simpson, Gavin (2018)) es muy útil para la visualización de los ajustes
 1145 de manera gráfica. Esta función toma los diagnósticos por defecto de la función **gam.check**
 1146 del paquete **mgcv** (Wood, Simon (2001)) y los convierte en gráficos de la librería **ggplot2**
 1147 (Wickham, Hadley(2011)) que son visualmente muy atractivos. La figura está dividida en
 1148 cuatro paneles, cada uno mostrando un diagnóstico diferente. En el panel superior izquierdo
 1149 se muestra un Q-Q Plot de los residuos. Si el modelo ha ajustado satisfactoriamente los
 1150 datos, los puntos deberían estar sobre la recta 1:1 demarcada en rojo. El panel superior
 1151 derecho muestra los residuos vs el predictor lineal. El objetivo de este diagnóstico es que los

1152 residuos se distribuyan al azar y que no tengan un patrón claro. La presencia de patrones
 1153 en los residuos indicaría que el modelo no ha explicado algún componente importante de
 1154 la variabilidad de los datos. En el panel inferior izquierdo se muestra un histograma de los
 1155 residuos siendo deseable que lo mismos tengan una distribución próxima a la Normal. El
 1156 último gráfico en el panel inferior derecho muestra una gráfica de dispersión entre los valores
 1157 ajustados y observados.

```

gratia:::appraise(gamgen_fit$fitted_models$tmax_fit) +
  ggplot2:::theme_bw()
  
```



1158

1159 Estos diagnósticos exploratorios pueden aplicarse a cada uno de los modelos ajustados por la

1160 función `spatial_calibrate`.
1161 Con la creación del objeto `gamgen_fit` finaliza el proceso de ajuste y ya se pueden generar
1162 series sintéticas para la o las estaciones usadas para calibrar el generador.

1163 **5.4.0.3 Generación de series** La generación de series sigue la misma estructura ante-
1164 rior, una función para configurar la generación y otra que realiza la generación propiamente
1165 dicha.

1166 Los argumentos de la función de control son:

- 1167 • `nsim`: cantidad de simulaciones a realizar. Se debe ingresar un valor **entero** mayor o
1168 igual a 1.
- 1169 • `seed`: semilla. Se debe ingresar cualquier numero **entero**. No es necesario recordarlo
1170 porque se guarda junto a los resultados.
- 1171 • `avbl_cores`: cantidad de núcleos disponibles para la paralelización.
- 1172 • `use_spatially_correlated_noise`: utilizar la generación estocástica espacialmente
1173 correlacionada. Esta opción sólo es válida si en el ajuste y en la simulación se usaron
1174 más de cinco estaciones meteorológicas diferentes. Con un menor número no es posible
1175 calcular los variogramas necesarios para la generación de los campos aleatorios. Se
1176 debe introducir un **boolean** (TRUE or FALSE).
- 1177 • `use_temporary_files_to_save_ram`: si se simulan muchas realizaciones o los recursos
1178 informáticos son escasos, esta opción permite guardar los resultados de cada una de las
1179 realizaciones en el disco liberando memoria RAM que quedará disponible para generar
1180 nuevas simulaciones. Al finalizar la generación todos los archivos se combinan en uno
1181 único. Se debe introducir un **boolean** (TRUE or FALSE).
- 1182 • `use_temporary_files_to_save_ram`: esta opción permite eliminar los archivos tem-
1183 porales creados para ahorrar RAM luego de terminar la generación de todas las simu-
1184 laciones. Se debe introducir un **boolean** (TRUE or FALSE).

```

control_sim <- gamwgen::spatial_simulation_control(
  nsim = 10, # Cantidad de simulaciones a realizar
  seed = 1234, # Semilla para que los resultados sean reproducibles
  avbl_cores = 6, # Cantidad de núcleos disponibles a utilizar
  use_spatially_correlated_noise = TRUE, # Usar modelo de ruido espacialmente correlacionado
  use_temporary_files_to_save_ram = TRUE, # Guardar resultados intermedios para ahorrar espacio
  remove_temp_files_used_to_save_ram = TRUE) # Borrar los resultados intermedios creados temporalmente

# Agregación de valores diarios
seasonal_covariates <- gamwgen::summarise_seasonal_climate(climate, umbral_faltantes = 0)

# Shapefile de Uruguay
uruguay <- raster::getData('GADM', country='URY', level=1) %>%
  sf::st_as_sf(.) %>%
  sf::st_transform(crs = 32721)

# Creación de la grilla sobre la que se simulará
grilla_simulacion_centers <- uruguay %>%
  dplyr::select(geometry) %>%
  sf::st_transform(gamgen_fit$crs_used_to_fit) %>%
  sf::st_make_grid(cellsize = 25000, what = 'centers') %>%
  sf::st_sf(grid_id = 1:length(.), geometry = .) %>%
  sf::st_intersection(uruguay)

1185 ## Warning: attribute variables are assumed to be spatially constant throughout all
1186 ## geometries

```

```

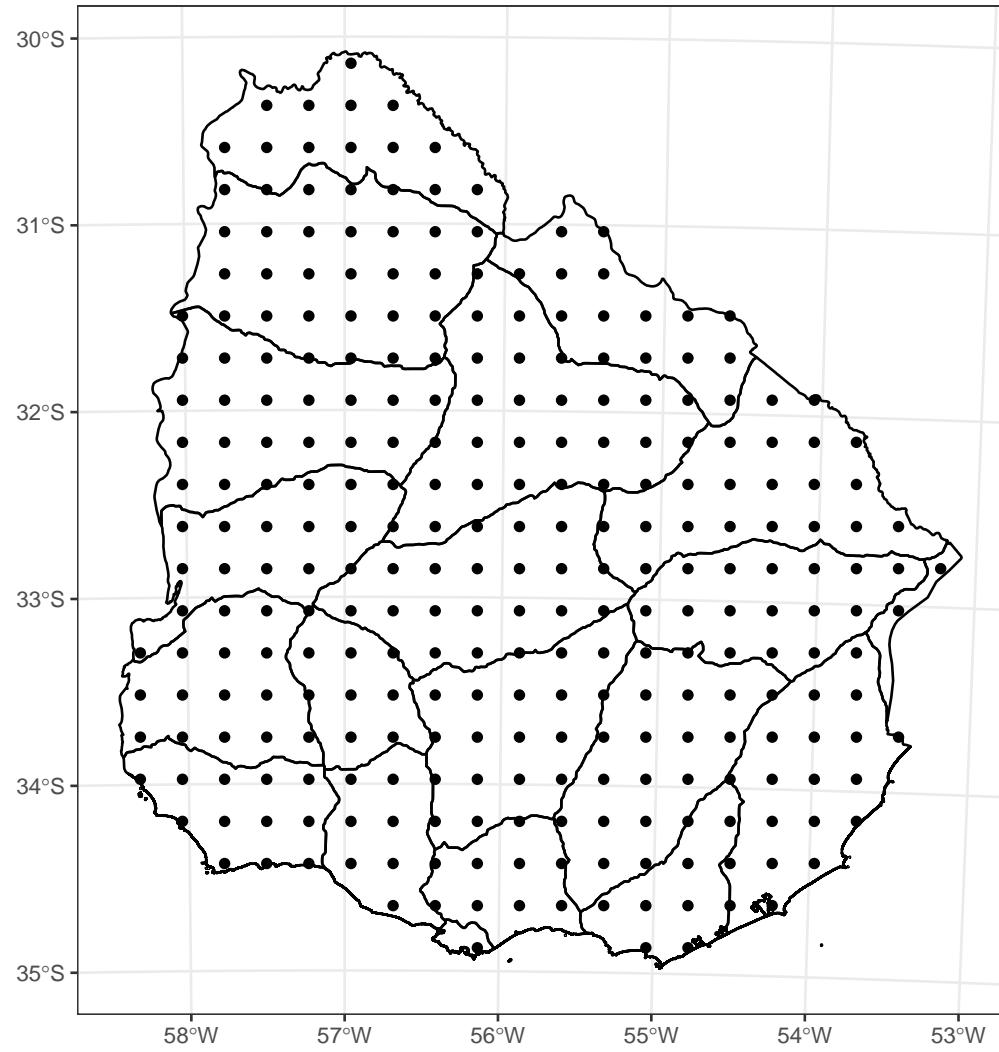
ggplot2::ggplot() +
  ggplot2::geom_sf(data = grilla_simulacion_centers) +

```

```

ggplot2::geom_sf(data = uruguay, fill = NA, color = "black", inherit.aes = FALSE) +
  ggplot2::theme_bw()

```



1187

1188 Al utilizar covariables trimestrales es necesario que éstas se interpolen para cada uno de
 1189 los puntos de la grilla. A continuación se muestran dos ejemplos, uno para precipitación
 1190 acumulada para el trimestre estival de 2019 y la temperatura máxima media del mismo
 1191 período.

```

simulation_dates <-
  tibble::tibble(date = seq.Date(from = as.Date('2019-01-01'),
  to = as.Date('2019-01-31')),

```

```

        by = "days")) %>%
dplyr::mutate(year = lubridate::year(date),
month = lubridate::month(date),
season = lubridate::quarter(date, fiscal_start = 12))

grilla_simulacion_centers <- uruguay %>%
sf::st_transform(gamgen_fit$crs_used_to_fit) %>%
sf::st_make_grid(cellsize = 25000, what = 'centers') %>%
sf::st_sf(grid_id = 1:length(.), geometry = .) %>%
sf::st_intersection(uruguay) %>%
dplyr::select(geometry) %>%
dplyr::mutate(latitude = sf::st_coordinates(.)[,2],
longitude = sf::st_coordinates(.)[,1])

seasonal_covariates <-
gamwgen:::get_covariates(model = gamgen_fit,
simulation_points = grilla_simulacion_centers,
seasonal_covariates,
simulation_dates,
control = control_sim)

aux <- gamwgen:::sf2raster(seasonal_covariates %>%
dplyr::filter(year == 2019, season == 1),
variable = 'seasonal_prcp') %>%
raster::as.data.frame(., xy = TRUE) %>%
dplyr::rename(., 'longitude' = 'x', 'latitude' = 'y') %>%

```

```

dplyr::mutate(., x = longitude, y = latitude) %>%
sf::st_as_sf(., coords = c('x', 'y')) %>%
sf::st_set_crs(., 32721) %>%
sf::st_intersection(uruguay) %>%
dplyr::select(latitude, longitude, z, geometry)

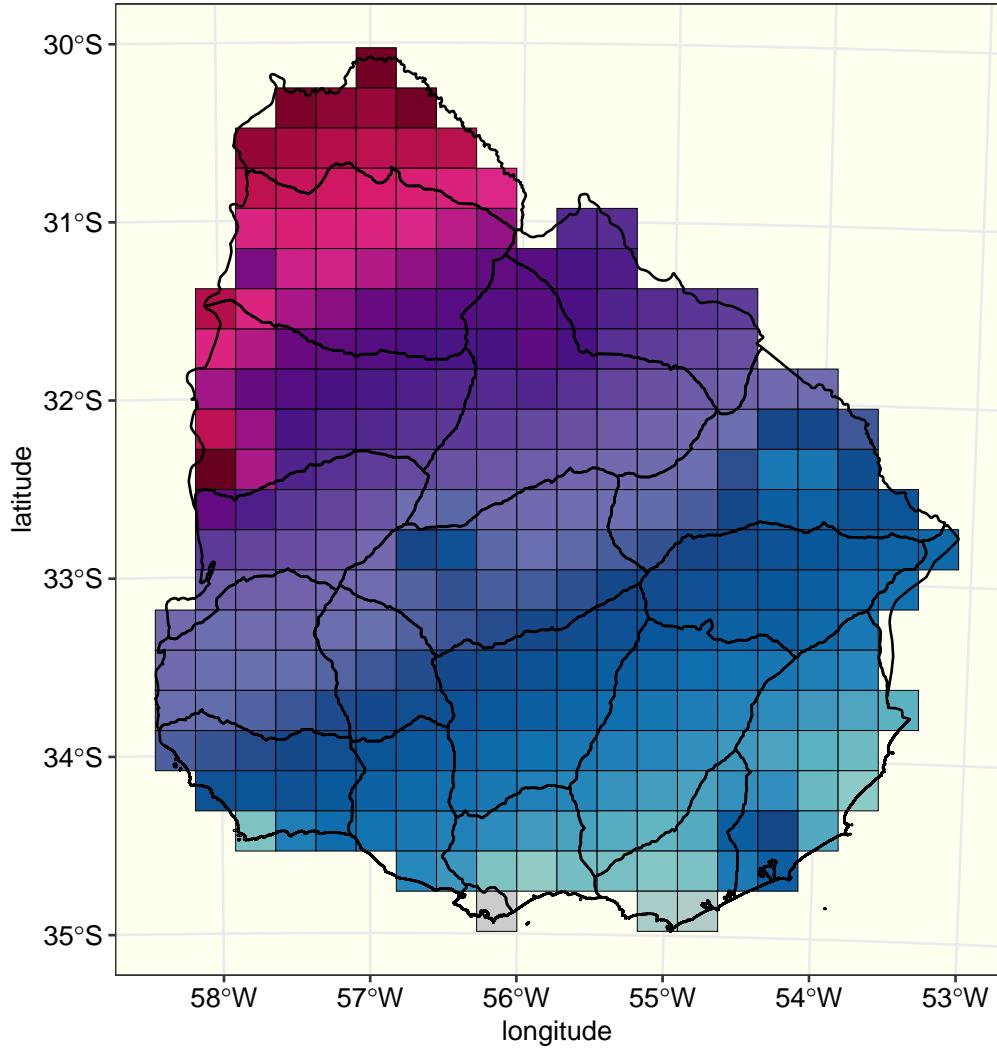
colores <- colorRampPalette(c("#cccccc", "#7bccc4", "#2b8cbe", "#0868ac",
                            "#084081", "#807dba", "#6a51a3", "#54278f", "#3f007d", "#e63399",
                            "#ce1256", "#67001f"))

quiebres <- c(0.0, 5.0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0, 90.0, 100.0, 110.0)

etiquetas <- c('0', '5', '10', '20', '30', '40', '50', '60', '70', '80', '90', '100', '110')

ggplot(data = aux,
        ggplot2::aes(x = longitude, y = latitude, fill = z)) +
  geom_tile(colour = "black") +
  scale_fill_gradientn(name = "Precipitation [mm]",
                        colours = colores(17),
                        breaks = quiebres,
                        labels = etiquetas) +
  ggplot2::geom_sf(data = uruguay, fill = NA, color = "black", inherit.aes = FALSE) +
  ggplot2::theme_bw() +
  ggplot2::theme(panel.background = element_rect(fill = "ivory"),
                axis.text = element_text(size = 11, colour = 1),
                legend.key.height = unit(2.5, "cm"))

```



1192

```

aux <- gamwgen:::sf2raster(seasonal_covariates %>%
  dplyr::filter(year == 2019, season == 1),
  variable = 'seasonal_tmax') %>%
raster::as.data.frame(., xy = TRUE) %>%
dplyr::rename(., 'longitude' = 'x', 'latitude' = 'y') %>%
dplyr::mutate(., x = longitude, y = latitude) %>%
sf::st_as_sf(., coords = c('x', 'y')) %>%
sf::st_set_crs(., 32721) %>%
sf::st_intersection(uruguay) %>%

```

```

dplyr::select(latitude, longitude, z, geometry)

1193 ## Warning: attribute variables are assumed to be spatially constant throughout all
1194 ## geometries

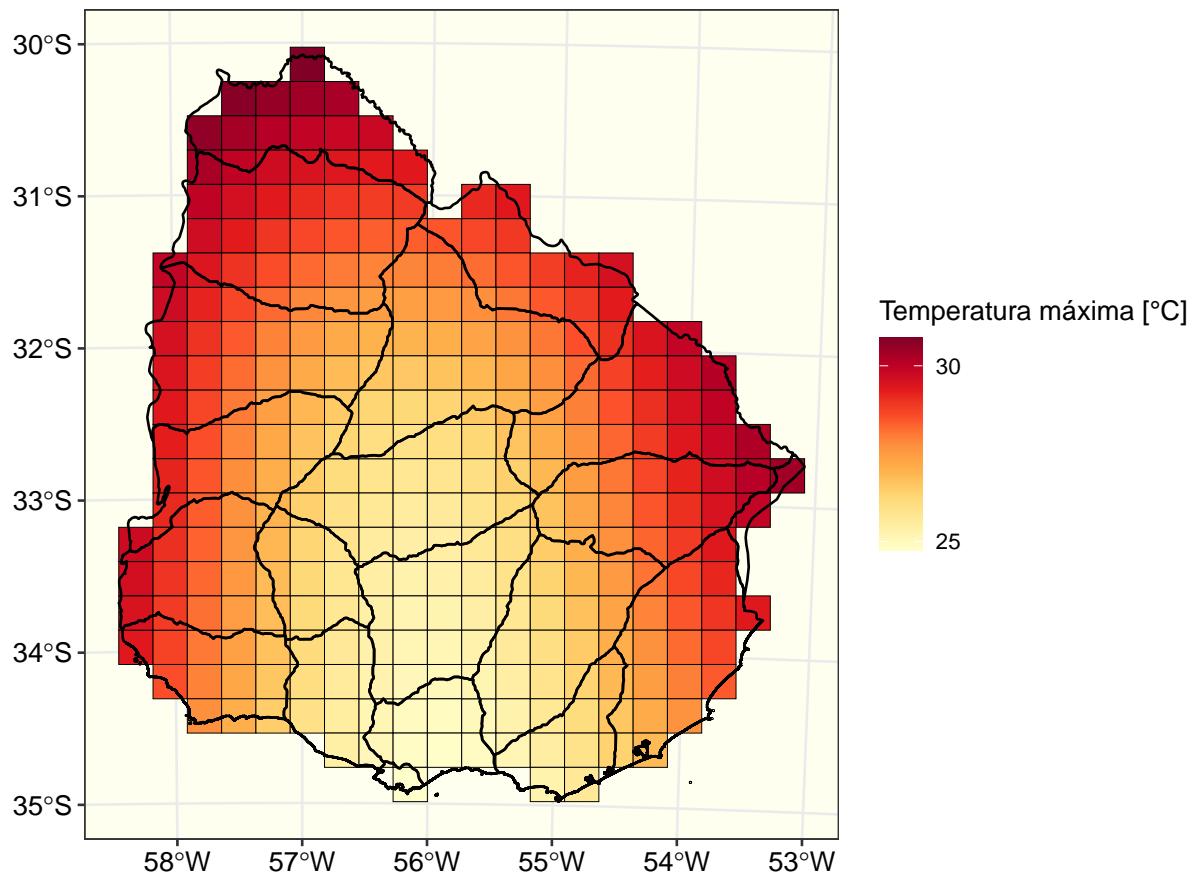
colr <- colorRampPalette(RColorBrewer::brewer.pal(9, 'YlOrRd'))

quiebres <- c(0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 45.0)

etiquetas <- c('0', '5', '10', '15', '20', '25', '30', '35', '40', '45')

ggplot(data = aux,
       ggplot2::aes(x = longitude, y = latitude, fill = z)) +
  geom_tile(colour="black") +
  scale_fill_gradientn(name = "Temperatura máxima [°C]",
                        colours = colr(9),
                        breaks = quiebres,
                        labels = etiquetas) +
  ggplot2::geom_sf(data = uruguay, fill = NA, color = "black", inherit.aes = FALSE) +
  ggplot2::theme_bw() +
  ggplot2:: labs(title = "", x = "", y = "") +
  ggplot2::theme(panel.background = element_rect(fill = "ivory"),
                 axis.text = element_text(size = 11, colour = 1))

```



1195

1196 Luego se procede a la simulación de datos meteorológicos. Los argumentos de la función de
 1197 simulación son:

- 1198 • **model**: objeto con el resultado de la función `local_calibrate()`
- 1199 • **simulation_locations**: objeto tipo `sf` con la ubicación de las estaciones a simu-
 1200 lar. Las estaciones usadas deben haber sido incluidas en el proceso de ajuste. No es
 1201 necesario que todas estén presentes, se pueden generar series solo sobre algunas de
 1202 ellas.
- 1203 • **start_date**: fecha de comienzo de la generación de series sintéticas. Si no se incluyeron
 1204 covariables estacionales en el ajuste, la fecha de comienzo es completamente arbitraria.

1205 Caso contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de
 1206 covariables, ni tampoco posterior. Se debe introducir una fecha en formato **date**
 1207
 • **end_date**: fecha de fin de la generación de series sintéticas. Si no se incluyeron co-
 1208 variables estacionales en el ajuste, la fecha de fin es completamente arbitraria. Caso
 1209 contrario, la fecha de comienzo no puede ser anterior al inicio de la serie de covariables,
 1210 tampoco puede ser posterior. Se debe introducir una fecha en formato **date**
 1211
 • **control**: objeto de control creado con la función **control_sim()**.
 1212
 • **output_folder**: ruta al directorio donde se guardarán los resultados, tanto finales
 1213 como intermedios.
 1214
 • **output_filename**: nombre del archivo de salida. Para facilitar la interoperabilidad,
 1215 el archivo generado es un archivo de texto en formato separado por comas (.csv)
 1216
 • **seasonal_covariates**: datos agregados trimestrales. Si el ajuste se realizó con co-
 1217 variables, la generación también debe realizarse con ellas. Caso contrario se producirá
 1218 un error. Se debe introducir un data frame con los valores agregados para las tres
 1219 variables (precipitación y temperaturas máxima y mínima) pero no necesariamente
 1220 deben ser los mismos a los utilizados en el ajuste. Si se desean simular tendencias de
 1221 algún tipo, ya sea de un modelo de cambio climático o arbitrarias, se deben perturbar
 1222 estas variables trimestrales e introducirlas aquí. Estas series si deben tener la misma
 1223 longitud que el período a generar
 1224
 • **verbose**: controla la impresión de mensajes en la consola. FALSE por defecto.

```

# Al correr la función se realiza la generación de series para cada una de las estacio-
# En este caso, por cuestiones de tiempo, vamos a cargar un objeto con los resultados
simulated_climate <- gamwgen::spatial_simulation(model = gamgen_fit, # Objeto con los r
  simulation_locations = grilla_simulacion_centers, # Estaciones para las cuales simu-
  start_date = as.Date('2019-01-01'), # Fecha de comienzo de las simulaciones
  end_date = as.Date('2019-01-31'), # Fecha de fin de las simulaciones
  control = control_sim, # Objeto con la configuración

```

```

    output_folder = getwd(), # Directorio donde se guardarán los resultados
    output_filename = 'simulations.csv', # Nombre del archivo de salida
    seasonal_covariates = seasonal_covariates, # Covariables estacionales
    verbose = FALSE) # Impresión de mensajes en la consola

```

1225 Esta función produce dos tipos de resultados: una lista que permanece en el ambiente de R y
 1226 los datos generados que son guardados como .csv en el directorio indicado precedentemente.

```

# Copiamos el archivo preajustado a nuestro directorio de trabajo
if (!fs::file_exists('output_data/spatial/simulated_spatial_conditional.RData')) {
  fs::file_copy(system.file('/autorun/spatial', "simulated_spatial_conditional.RData",
                           new_path = 'output_data/spatial/simulated_spatial_conditional.RData')
}
# Cargamos el archivo recientemente creado
load('output_data/spatial/simulated_spatial_conditional.RData')

# Clase del objeto con el ajuste del generador
class(simulated_climate)

1227 ## [1] "list"           "gamwgen.climate"

# Contenido del modelo
names(simulated_climate)

1228 ## [1] "nsim"
1229 ## [2] "seed"
1230 ## [3] "realizations_seeds"
1231 ## [4] "simulation_points"
1232 ## [5] "output_file_with_results"

```

```
1233 ## [6] "output_file_fomart"  
1234 ## [7] "rdata_file_with_fitted_stations_and_climate"  
1235 ## [8] "exec_times"
```

1236 La lista contiene los siguientes objetos:

- 1237 • `nsim`: cantidad de realizaciones.
- 1238 • `seed`: semilla general para toda la generación. Corresponde a la que se incluye en la
1239 función de control.
- 1240 • `realization_seeds`: semillas para cada una de las realizaciones. Esto permite replicar
1241 los resultados.
- 1242 • `simulation_points`: puntos donde se generaron las series sintéticas.
- 1243 • `output_file_with_results`: nombre del archivo con los resultados.
- 1244 • `output_file_format`: tipo de archivo de salida, en este caso `.csv`.
- 1245 • `rdata_file_with_fitted_stations_and_climate`: archivo `.RData` con los datos me-
1246 teorológicos observados que fueron utilizados en el ajuste. También se incluyen los
1247 metadatos de cada uno de esos puntos.
- 1248 • `exec_times`: tiempo de ejecución de la generación.

1249 Ahora veremos el formato del archivo de salida que contiene las series sintéticas.

1250 Para desmenuzar la generación del tiempo local se pueden correr algunas de las funciones
1251 que forman parte de `local_simulation`. Por ejemplo, del objeto `gamgen_fit` se extraen
1252 los residuos y con la función `gamwgen:::generate_residuals_statistics` se calculan los
1253 parámetros.

1254 En la tabla anterior se muestran los parámetros necesarios para generar el tiempo local de
1255 las temperaturas máxima y mínima.

```
gen_noise_params <- gamwgen:::generate_month_params(  
  residuals = gamgen_fit$models_residuals,
```

```

observed_climate = gamgen_fit$models_data,
stations = gamgen_fit$stations)

# Ejemplo de variograma de residuos de temperatura máxima para días secos

gen_noise_params[[1]]$variogram_parameters$tmax_dry

1256 ## [1] 0.000000 7.045242 493.298928

1257 Explicar variograma.

1258 Los parámetros para cada uno de los meses permiten generar valores de tiempo local para
1259 las dos temperaturas a partir de una distribución normal multivariada.

1260 A continuación se muestra un ejemplo para el año 2019 sólo para los días lluviosos.

```

```
RandomFields::RFoptions(printlevel = 0, spConform = FALSE)
```

```
# Mapa de Uruguay
```

```
uruguay <- raster::getData('GADM', country='URY', level=1) %>%
  sf::st_as_sf(.) %>%
  sf::st_transform(crs = 32721)
```

```
# Grilla de simulación
```

```
grilla_simulacion_centers <- uruguay %>%
  sf::st_transform(gamgen_fit$crs_used_to_fit) %>%
  sf::st_make_grid(cellsize = 25000, what = 'centers') %>%
  sf::st_sf(grid_id = 1:length(.), geometry = .) %>%
  sf::st_intersection(uruguay) %>%
  dplyr::select(geometry) %>%
```

```

dplyr::mutate(latitude = sf::st_coordinates(.)[,2],
               longitude = sf::st_coordinates(.)[,1])

temp_local    <- control_sim$temperature_noise_generating_function(
  simulation_points =  grilla_simulacion_centers,
  gen_noise_params = gen_noise_params,
  month_number = 1L,
  selector = 'Dry',
  seed = 1234) %>%
  dplyr::mutate(date = as.Date(paste0('2019-', '01', '-01'))) %>%
  dplyr::filter(date == as.Date('2019-01-01')) %>%
  tidyrr::gather(variable, valor, -c('geometry'))

tmax_residuals <- gamwgen:::sf2raster(temp_local %>%
                                         dplyr::filter(variable == 'tmax_residuals'),
                                         'valor') %>%
  raster::as.data.frame(., xy = TRUE) %>%
  dplyr::rename(., 'longitude' = 'x', 'latitude' = 'y') %>%
  dplyr::mutate(., x = longitude, y = latitude) %>%
  sf::st_as_sf(., coords = c('x', 'y')) %>%
  sf::st_set_crs(., 32721) %>%
  sf::st_intersection(uruguay) %>%
  dplyr::select(latitude, longitude, z, geometry)

colr <- colorRampPalette(RColorBrewer::brewer.pal(9, 'YlOrRd'))

```

```

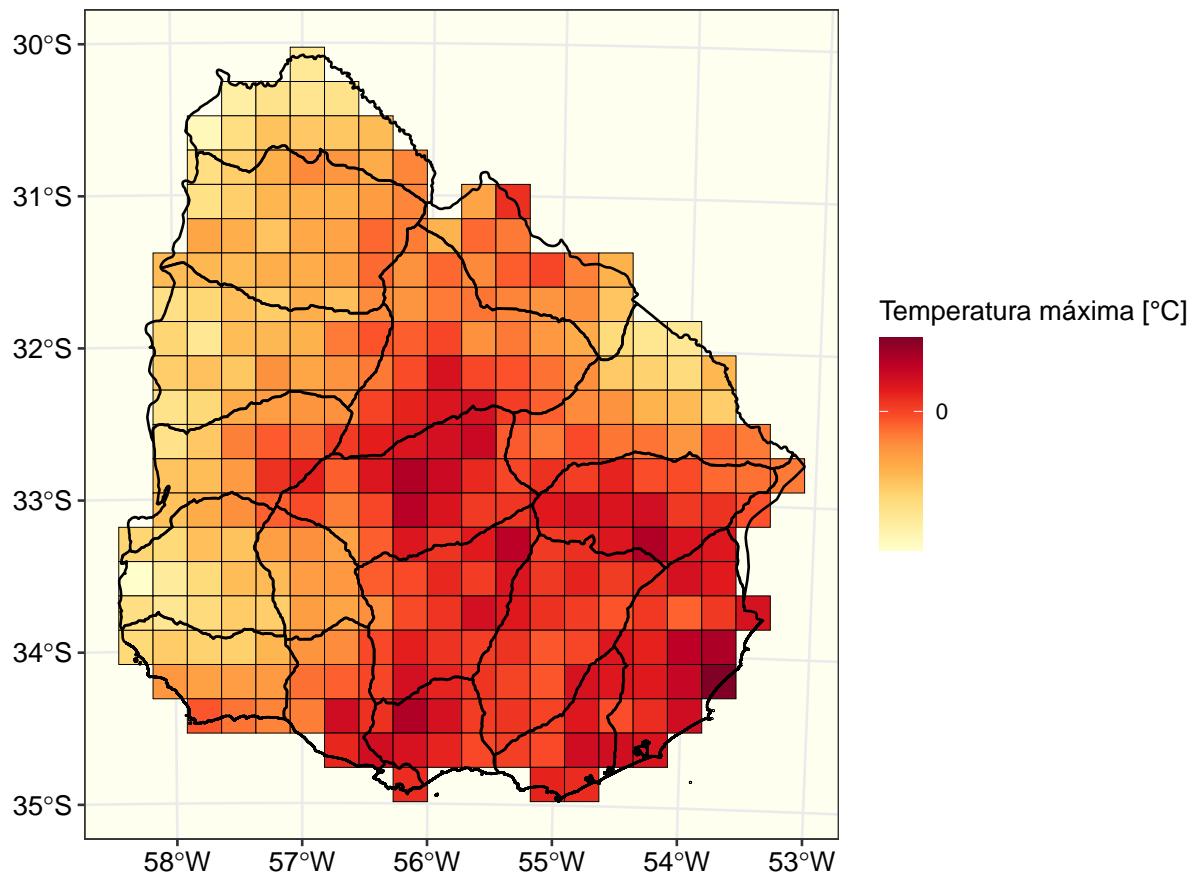
quiebres <- c(0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 45.0)

etiquetas <- c('0', '5', '10', '15', '20', '25', '30', '35', '40', '45')

ggplot(data = tmax_residuals,
        ggplot2::aes(x = longitude, y = latitude, fill = z)) +
  geom_tile(colour="black") +
  scale_fill_gradientn(name = "Temperatura máxima [°C]",
                        colours = colr(9),
                        breaks = quiebres,
                        labels = etiquetas) +
  ggplot2::geom_sf(data = uruguay, fill = NA, color = "black", inherit.aes = FALSE) +
  ggplot2::theme_bw() +
  ggplot2:: labs(title = "", x = "", y = "") +
  ggplot2::theme(panel.background = element_rect(fill = "ivory"),
                 axis.text = element_text(size = 11, colour = 1))

```

1261



1262

1263 El mapa muestra el campo Gaussiano de temperatura máxima para el primer día de la
1264 simulación, 1 de enero de 2019.

Se carga el set de datos simulados

```
simulated_climate <- readr::read_csv(here::here('output_data/spatial/simulated_spatial_0'))
```

Primeras filas del objeto de salidas

```
knitr::kable(head(simulated_climate), "latex", booktabs = T) %>%
```

```
  kableExtra::kable_styling(position = "center",
    latex_options = c("striped", "scale_down"))
```

realization	point_id	longitude	latitude	date	tmax	tmin	prcp
1	1	453759.4	6590418	2019-01-01	33.03589	19.72844	1.826571
1	2	553759.4	6590418	2019-01-01	31.77257	19.65962	27.883864
1	3	578759.4	6590418	2019-01-01	32.71455	20.47479	32.879767
1	4	428759.4	6615418	2019-01-01	30.64363	16.02136	7.032041
1	5	453759.4	6615418	2019-01-01	31.83107	18.12258	14.671730
1	6	478759.4	6615418	2019-01-01	33.32291	18.75998	14.781878

1265 El resultado de la generación es un archivo .csv que contiene la siguiente información:

- 1266 • **realization**: número de realización. Es un valor entero entre 1 y la cantidad de
- 1267 realizaciones definida por el usuario.
- 1268 • **station_id**: número único de identificación de la estación meteorológica o del punto
- 1269 arbitrario.
- 1270 • **date**: fechas de cada uno de los días de la simulación.
- 1271 • **tmax**: valores de temperatura máxima generada expresada en °C.
- 1272 • **tmin**: valores de temperatura mínima generada expresada en °C.
- 1273 • **prcp**: valores de precipitación diaria generada expresada en mm.

1274 La siguiente Figura muestra un ejemplo de las series de temperaturas máximas y mínimas
1275 generadas.

```
uruguay <- raster::getData('GADM', country='URY', level=1) %>%
  sf::st_as_sf(.) %>%
  sf::st_transform(crs = 32721)

simulated_climate_geo <- simulated_climate %>%
```

```

dplyr::filter(date == as.Date('2019-01-01'),
              realization == 1) %>%
sf::st_as_sf(coords = c('longitude', 'latitude'), crs = 32721) %>%
gamwgen:::sf2raster(., 'prcp') %>%
raster::as.data.frame(., xy = TRUE) %>%
dplyr::rename(., 'longitude' = 'x', 'latitude' = 'y') %>%
dplyr::mutate(., x = longitude, y = latitude) %>%
sf::st_as_sf(., coords = c('x', 'y')) %>%
sf::st_set_crs(., 32721) %>%
sf::st_intersection(uruguay) %>%
dplyr::select(latitude, longitude, z, geometry)

1276 ## Warning: attribute variables are assumed to be spatially constant throughout all
1277 ## geometries

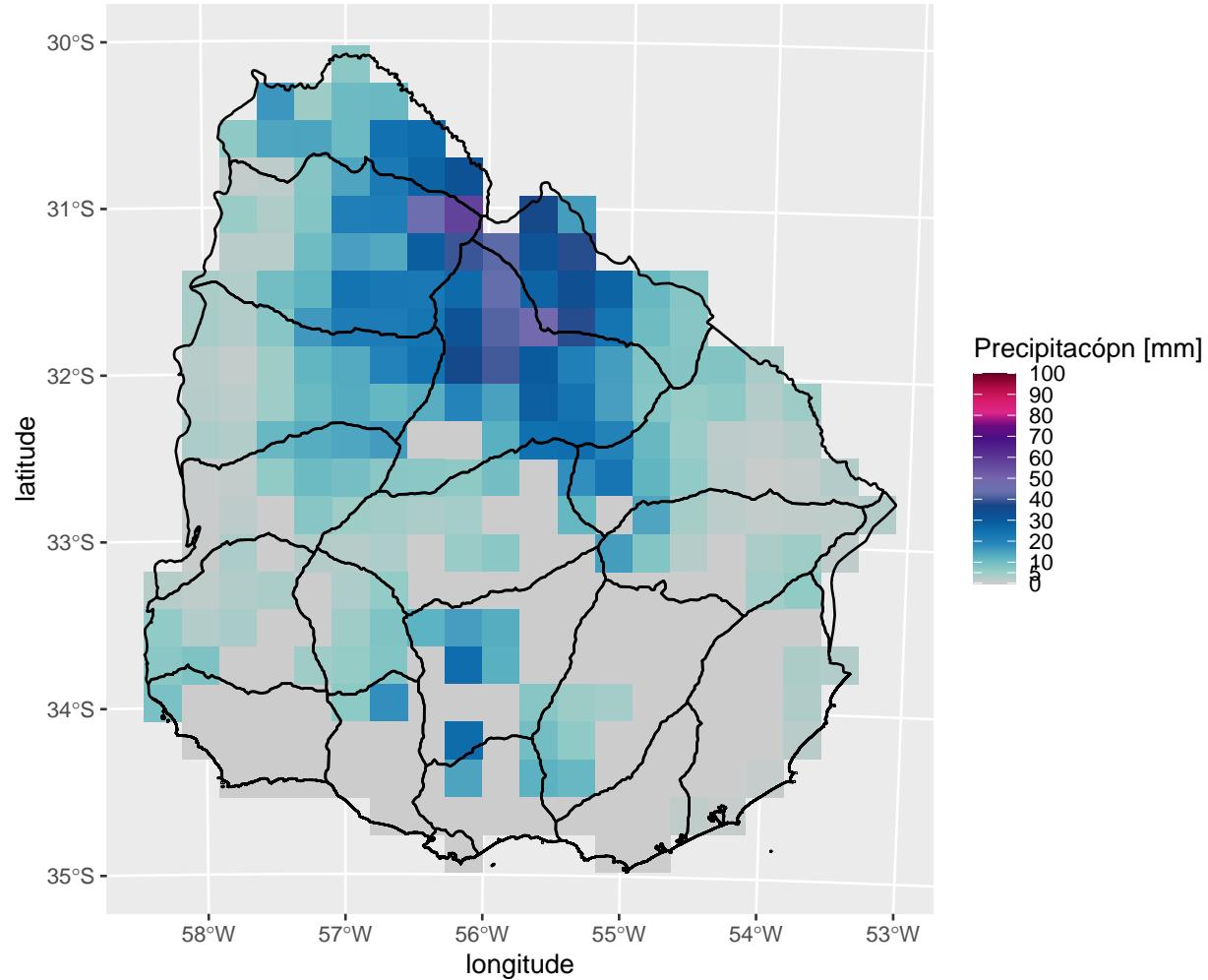
colores <- colorRampPalette(c("#cccccc", "#7bccc4", "#2b8cbe", "#0868ac",
                            "#084081", "#807dba", "#6a51a3", "#54278f", "#3f007d", "#e7
                            "#ce1256", "#67001f"))

quiebres <- c(0.0, 5.0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0, 90.0, 100.0, 110.0, 120.0, 130.0, 140.0, 150.0, 160.0, 170.0, 180.0, 190.0, 200.0, 210.0, 220.0, 230.0, 240.0, 250.0, 260.0, 270.0, 280.0, 290.0, 300.0, 310.0, 320.0, 330.0, 340.0, 350.0, 360.0, 370.0, 380.0, 390.0, 400.0, 410.0, 420.0, 430.0, 440.0, 450.0, 460.0, 470.0, 480.0, 490.0, 500.0, 510.0, 520.0, 530.0, 540.0, 550.0, 560.0, 570.0, 580.0, 590.0, 600.0, 610.0, 620.0, 630.0, 640.0, 650.0, 660.0, 670.0, 680.0, 690.0, 700.0, 710.0, 720.0, 730.0, 740.0, 750.0, 760.0, 770.0, 780.0, 790.0, 800.0, 810.0, 820.0, 830.0, 840.0, 850.0, 860.0, 870.0, 880.0, 890.0, 900.0, 910.0, 920.0, 930.0, 940.0, 950.0, 960.0, 970.0, 980.0, 990.0, 1000.0, 1010.0, 1020.0, 1030.0, 1040.0, 1050.0, 1060.0, 1070.0, 1080.0, 1090.0, 1100.0, 1110.0, 1120.0, 1130.0, 1140.0, 1150.0, 1160.0, 1170.0, 1180.0, 1190.0, 1200.0, 1210.0, 1220.0, 1230.0, 1240.0, 1250.0, 1260.0, 1270.0, 1280.0, 1290.0, 1300.0, 1310.0, 1320.0, 1330.0, 1340.0, 1350.0, 1360.0, 1370.0, 1380.0, 1390.0, 1400.0, 1410.0, 1420.0, 1430.0, 1440.0, 1450.0, 1460.0, 1470.0, 1480.0, 1490.0, 1500.0, 1510.0, 1520.0, 1530.0, 1540.0, 1550.0, 1560.0, 1570.0, 1580.0, 1590.0, 1600.0, 1610.0, 1620.0, 1630.0, 1640.0, 1650.0, 1660.0, 1670.0, 1680.0, 1690.0, 1700.0, 1710.0, 1720.0, 1730.0, 1740.0, 1750.0, 1760.0, 1770.0, 1780.0, 1790.0, 1800.0, 1810.0, 1820.0, 1830.0, 1840.0, 1850.0, 1860.0, 1870.0, 1880.0, 1890.0, 1900.0, 1910.0, 1920.0, 1930.0, 1940.0, 1950.0, 1960.0, 1970.0, 1980.0, 1990.0, 2000.0, 2010.0, 2020.0, 2030.0, 2040.0, 2050.0, 2060.0, 2070.0, 2080.0, 2090.0, 2100.0, 2110.0, 2120.0, 2130.0, 2140.0, 2150.0, 2160.0, 2170.0, 2180.0, 2190.0, 2200.0, 2210.0, 2220.0, 2230.0, 2240.0, 2250.0, 2260.0, 2270.0, 2280.0, 2290.0, 2300.0, 2310.0, 2320.0, 2330.0, 2340.0, 2350.0, 2360.0, 2370.0, 2380.0, 2390.0, 2400.0, 2410.0, 2420.0, 2430.0, 2440.0, 2450.0, 2460.0, 2470.0, 2480.0, 2490.0, 2500.0, 2510.0, 2520.0, 2530.0, 2540.0, 2550.0, 2560.0, 2570.0, 2580.0, 2590.0, 2600.0, 2610.0, 2620.0, 2630.0, 2640.0, 2650.0, 2660.0, 2670.0, 2680.0, 2690.0, 2700.0, 2710.0, 2720.0, 2730.0, 2740.0, 2750.0, 2760.0, 2770.0, 2780.0, 2790.0, 2800.0, 2810.0, 2820.0, 2830.0, 2840.0, 2850.0, 2860.0, 2870.0, 2880.0, 2890.0, 2900.0, 2910.0, 2920.0, 2930.0, 2940.0, 2950.0, 2960.0, 2970.0, 2980.0, 2990.0, 3000.0, 3010.0, 3020.0, 3030.0, 3040.0, 3050.0, 3060.0, 3070.0, 3080.0, 3090.0, 3100.0, 3110.0, 3120.0, 3130.0, 3140.0, 3150.0, 3160.0, 3170.0, 3180.0, 3190.0, 3200.0, 3210.0, 3220.0, 3230.0, 3240.0, 3250.0, 3260.0, 3270.0, 3280.0, 3290.0, 3300.0, 3310.0, 3320.0, 3330.0, 3340.0, 3350.0, 3360.0, 3370.0, 3380.0, 3390.0, 3400.0, 3410.0, 3420.0, 3430.0, 3440.0, 3450.0, 3460.0, 3470.0, 3480.0, 3490.0, 3500.0, 3510.0, 3520.0, 3530.0, 3540.0, 3550.0, 3560.0, 3570.0, 3580.0, 3590.0, 3600.0, 3610.0, 3620.0, 3630.0, 3640.0, 3650.0, 3660.0, 3670.0, 3680.0, 3690.0, 3700.0, 3710.0, 3720.0, 3730.0, 3740.0, 3750.0, 3760.0, 3770.0, 3780.0, 3790.0, 3800.0, 3810.0, 3820.0, 3830.0, 3840.0, 3850.0, 3860.0, 3870.0, 3880.0, 3890.0, 3900.0, 3910.0, 3920.0, 3930.0, 3940.0, 3950.0, 3960.0, 3970.0, 3980.0, 3990.0, 4000.0, 4010.0, 4020.0, 4030.0, 4040.0, 4050.0, 4060.0, 4070.0, 4080.0, 4090.0, 4100.0, 4110.0, 4120.0, 4130.0, 4140.0, 4150.0, 4160.0, 4170.0, 4180.0, 4190.0, 4200.0, 4210.0, 4220.0, 4230.0, 4240.0, 4250.0, 4260.0, 4270.0, 4280.0, 4290.0, 4300.0, 4310.0, 4320.0, 4330.0, 4340.0, 4350.0, 4360.0, 4370.0, 4380.0, 4390.0, 4400.0, 4410.0, 4420.0, 4430.0, 4440.0, 4450.0, 4460.0, 4470.0, 4480.0, 4490.0, 4500.0, 4510.0, 4520.0, 4530.0, 4540.0, 4550.0, 4560.0, 4570.0, 4580.0, 4590.0, 4600.0, 4610.0, 4620.0, 4630.0, 4640.0, 4650.0, 4660.0, 4670.0, 4680.0, 4690.0, 4700.0, 4710.0, 4720.0, 4730.0, 4740.0, 4750.0, 4760.0, 4770.0, 4780.0, 4790.0, 4800.0, 4810.0, 4820.0, 4830.0, 4840.0, 4850.0, 4860.0, 4870.0, 4880.0, 4890.0, 4900.0, 4910.0, 4920.0, 4930.0, 4940.0, 4950.0, 4960.0, 4970.0, 4980.0, 4990.0, 5000.0, 5010.0, 5020.0, 5030.0, 5040.0, 5050.0, 5060.0, 5070.0, 5080.0, 5090.0, 5100.0, 5110.0, 5120.0, 5130.0, 5140.0, 5150.0, 5160.0, 5170.0, 5180.0, 5190.0, 5200.0, 5210.0, 5220.0, 5230.0, 5240.0, 5250.0, 5260.0, 5270.0, 5280.0, 5290.0, 5300.0, 5310.0, 5320.0, 5330.0, 5340.0, 5350.0, 5360.0, 5370.0, 5380.0, 5390.0, 5400.0, 5410.0, 5420.0, 5430.0, 5440.0, 5450.0, 5460.0, 5470.0, 5480.0, 5490.0, 5500.0, 5510.0, 5520.0, 5530.0, 5540.0, 5550.0, 5560.0, 5570.0, 5580.0, 5590.0, 5600.0, 5610.0, 5620.0, 5630.0, 5640.0, 5650.0, 5660.0, 5670.0, 5680.0, 5690.0, 5700.0, 5710.0, 5720.0, 5730.0, 5740.0, 5750.0, 5760.0, 5770.0, 5780.0, 5790.0, 5800.0, 5810.0, 5820.0, 5830.0, 5840.0, 5850.0, 5860.0, 5870.0, 5880.0, 5890.0, 5900.0, 5910.0, 5920.0, 5930.0, 5940.0, 5950.0, 5960.0, 5970.0, 5980.0, 5990.0, 6000.0, 6010.0, 6020.0, 6030.0, 6040.0, 6050.0, 6060.0, 6070.0, 6080.0, 6090.0, 6100.0, 6110.0, 6120.0, 6130.0, 6140.0, 6150.0, 6160.0, 6170.0, 6180.0, 6190.0, 6200.0, 6210.0, 6220.0, 6230.0, 6240.0, 6250.0, 6260.0, 6270.0, 6280.0, 6290.0, 6300.0, 6310.0, 6320.0, 6330.0, 6340.0, 6350.0, 6360.0, 6370.0, 6380.0, 6390.0, 6400.0, 6410.0, 6420.0, 6430.0, 6440.0, 6450.0, 6460.0, 6470.0, 6480.0, 6490.0, 6500.0, 6510.0, 6520.0, 6530.0, 6540.0, 6550.0, 6560.0, 6570.0, 6580.0, 6590.0, 6600.0, 6610.0, 6620.0, 6630.0, 6640.0, 6650.0, 6660.0, 6670.0, 6680.0, 6690.0, 6700.0, 6710.0, 6720.0, 6730.0, 6740.0, 6750.0, 6760.0, 6770.0, 6780.0, 6790.0, 6800.0, 6810.0, 6820.0, 6830.0, 6840.0, 6850.0, 6860.0, 6870.0, 6880.0, 6890.0, 6900.0, 6910.0, 6920.0, 6930.0, 6940.0, 6950.0, 6960.0, 6970.0, 6980.0, 6990.0, 7000.0, 7010.0, 7020.0, 7030.0, 7040.0, 7050.0, 7060.0, 7070.0, 7080.0, 7090.0, 7100.0, 7110.0, 7120.0, 7130.0, 7140.0, 7150.0, 7160.0, 7170.0, 7180.0, 7190.0, 7200.0, 7210.0, 7220.0, 7230.0, 7240.0, 7250.0, 7260.0, 7270.0, 7280.0, 7290.0, 7300.0, 7310.0, 7320.0, 7330.0, 7340.0, 7350.0, 7360.0, 7370.0, 7380.0, 7390.0, 7400.0, 7410.0, 7420.0, 7430.0, 7440.0, 7450.0, 7460.0, 7470.0, 7480.0, 7490.0, 7500.0, 7510.0, 7520.0, 7530.0, 7540.0, 7550.0, 7560.0, 7570.0, 7580.0, 7590.0, 7600.0, 7610.0, 7620.0, 7630.0, 7640.0, 7650.0, 7660.0, 7670.0, 7680.0, 7690.0, 7700.0, 7710.0, 7720.0, 7730.0, 7740.0, 7750.0, 7760.0, 7770.0, 7780.0, 7790.0, 7800.0, 7810.0, 7820.0, 7830.0, 7840.0, 7850.0, 7860.0, 7870.0, 7880.0, 7890.0, 7900.0, 7910.0, 7920.0, 7930.0, 7940.0, 7950.0, 7960.0, 7970.0, 7980.0, 7990.0, 8000.0, 8010.0, 8020.0, 8030.0, 8040.0, 8050.0, 8060.0, 8070.0, 8080.0, 8090.0, 8100.0, 8110.0, 8120.0, 8130.0, 8140.0, 8150.0, 8160.0, 8170.0, 8180.0, 8190.0, 8200.0, 8210.0, 8220.0, 8230.0, 8240.0, 8250.0, 8260.0, 8270.0, 8280.0, 8290.0, 8300.0, 8310.0, 8320.0, 8330.0, 8340.0, 8350.0, 8360.0, 8370.0, 8380.0, 8390.0, 8400.0, 8410.0, 8420.0, 8430.0, 8440.0, 8450.0, 8460.0, 8470.0, 8480.0, 8490.0, 8500.0, 8510.0, 8520.0, 8530.0, 8540.0, 8550.0, 8560.0, 8570.0, 8580.0, 8590.0, 8600.0, 8610.0, 8620.0, 8630.0, 8640.0, 8650.0, 8660.0, 8670.0, 8680.0, 8690.0, 8700.0, 8710.0, 8720.0, 8730.0, 8740.0, 8750.0, 8760.0, 8770.0, 8780.0, 8790.0, 8800.0, 8810.0, 8820.0, 8830.0, 8840.0, 8850.0, 8860.0, 8870.0, 8880.0, 8890.0, 8890.0, 8900.0, 8910.0, 8920.0, 8930.0, 8940.0, 8950.0, 8960.0, 8970.0, 8980.0, 8990.0, 9000.0, 9010.0, 9020.0, 9030.0, 9040.0, 9050.0, 9060.0, 9070.0, 9080.0, 9090.0, 9100.0, 9110.0, 9120.0, 9130.0, 9140.0, 9150.0, 9160.0, 9170.0, 9180.0, 9190.0, 9200.0, 9210.0, 9220.0, 9230.0, 9240.0, 9250.0, 9260.0, 9270.0, 9280.0, 9290.0, 9300.0, 9310.0, 9320.0, 9330.0, 9340.0, 9350.0, 9360.0, 9370.0, 9380.0, 9390.0, 9400.0, 9410.0, 9420.0, 9430.0, 9440.0, 9450.0, 9460.0, 9470.0, 9480.0, 9490.0, 9500.0, 9510.0, 9520.0, 9530.0, 9540.0, 9550.0, 9560.0, 9570.0, 9580.0, 9590.0, 9600.0, 9610.0, 9620.0, 9630.0, 9640.0, 9650.0, 9660.0, 9670.0, 9680.0, 9690.0, 9700.0, 9710.0, 9720.0, 9730.0, 9740.0, 9750.0, 9760.0, 9770.0, 9780.0, 9790.0, 9800.0, 9810.0, 9820.0, 9830.0, 9840.0, 9850.0, 9860.0, 9870.0, 9880.0, 9890.0, 9890.0, 9900.0, 9910.0, 9920.0, 9930.0, 9940.0, 9950.0, 9960.0, 9970.0, 9980.0, 9990.0, 9990.0, 10000.0, 10010.0, 10020.0, 10030.0, 10040.0, 10050.0, 10060.0, 10070.0, 10080.0, 10090.0, 10090.0, 10100.0, 10110.0, 10120.0, 10130.0, 10140.0, 10150.0, 10160.0, 10170.0, 10180.0, 10190.0, 10190.0, 10200.0, 10210.0, 10220.0, 10230.0, 10240.0, 10250.0, 10260.0, 10270.0, 10280.0, 10290.0, 10290.0, 10300.0, 10310.0, 10320.0, 10330.0, 10340.0, 10350.0, 10360.0, 10370.0, 10380.0, 10390.0, 10390.0, 10400.0, 10410.0, 10420.0, 10430.0, 10440.0, 10450.0, 10460.0, 10470.0, 10480.0, 10490.0, 10490.0, 10500.0, 10510.0, 10520.0, 10530.0, 10540.0, 10550.0, 10560.0, 10570.0, 10580.0, 10590.0, 10590.0, 10600.0, 10610.0, 10620.0, 10630.0, 10640.0, 10650.0, 10660.0, 10670.0, 10680.0, 10690.0, 10690.0, 10700.0, 10710.0, 10720.0, 10730.0, 10740.0, 10750.0, 10760.0, 10770.0, 10780.0, 10790.0, 10790.0, 10800.0, 10810.0, 10820.0, 10830.0, 10840.0, 10850.0, 10860.0, 10870.0, 10880.0, 10890.0, 10890.0, 10900.0, 10910.0, 10920.0, 10930.0, 10940.0, 10950.0, 10960.0, 10970.0, 10980.0, 10990.0, 10990.0, 11000.0, 11010.0, 11020.0, 11030.0, 11040.0, 11050.0, 11060.0, 11070.0, 11080.0, 11090.0, 11090.0, 11100.0, 11110.0, 11120.0, 11130.0, 11140.0, 11150.0, 11160.0, 11170.0, 11180.0, 11190.0, 11190.0, 11200.0, 11210.0, 11220.0, 11230.0, 11240.0, 11250.0, 11260.0, 11270.0, 11280.0, 11290.0, 11290.0, 11300.0, 11310.0, 11320.0, 11330.0, 11340.0, 11350.0, 11360.0, 11370.0, 11380.0, 11390.0, 11390.0, 11400.0, 11410.0, 11420.0, 11430.0, 11440.0, 11450.0, 11460.0, 11470.0, 11480.0, 11490.0, 11490.0, 11500.0, 11510.0, 11520.0, 11530.0, 11540.0, 11550.0, 11560.0, 11570.0, 11580.0, 11590.0, 11590.0, 11600.0, 11610.0, 11620.0, 11630.0, 11640.0, 11650.0, 11660.0, 11670.0, 11680.0, 11690.0, 11690.0, 11700.0, 11710.0, 11720.0, 11730.0, 11740.0, 11750.0, 11760.0, 11770.0, 11780.0, 11790.0, 11790.0, 11800.0, 11810.0, 11820.0, 11830.0, 11840.0, 11850.0, 11860.0, 11870.0, 11880.0, 11890.0, 11890.0, 11900.0, 11910.0, 11920.0, 11930.0, 11940.0, 11950.0, 11960.0, 11970.0, 11980.0, 11990.0, 11990.0, 12000.0, 12010.0, 12020.0, 12030.0, 12040.0, 12050.0, 12060.0, 12070.0, 12080.0, 12090.0, 12090.0, 12100.0, 12110.0, 12120.0, 12130.0, 12140.0, 12150.0, 12160.0, 12170.0, 12180.0, 12190.0, 12190.0, 12200.0, 12210.0, 12220.0, 12230.0, 12240.0, 12250.0, 12260.0, 12270.0, 12280.0, 12290.0, 12290.0, 12300.0, 12310.0, 12320.0, 12330.0, 12340.0, 12350.0, 12360.0, 12370.0, 12380.0, 12390.0, 12390.0, 12400.0, 12410.0, 12420.0, 12430.0, 12440.0, 12450.0, 12460.0, 12470.0, 12480.0, 12490.0, 12490.0, 12500.0, 12510.0, 12520.0, 12530.0, 12540.0, 12550.0, 12560.0, 12570.0, 12580.0, 12590.0, 12590.0, 12600.0, 12610.0, 12620.0, 12630.0, 12640.0, 12650.0, 12660.0, 12670.0, 12680.0, 12690.0, 12690.0, 12700.0, 12710.0, 12720.0, 12730.0, 12740.0, 12750.0, 12760.0, 12770.0, 12780.0, 12790.0, 12790.0, 12800.0, 12810.0, 12820.0, 12830.0, 12840.0, 12850.0, 12860.0, 12870.0, 12880.0, 12890.0, 12890.0, 12900.0, 12910.0, 12920.0, 12930.0, 12940.0, 12950.0, 12960.0, 12970.0, 12980.0, 12990.0, 12990.0, 13000.0, 13010.0, 13020.0, 13030.0, 13040.0, 13050.0, 13060.0, 13070.0, 13080.0, 13090.0, 13090.0, 13100.0, 13110.0, 13120.0, 13130.0, 13
```

```

      breaks = quiebres,
      labels = etiquetas,
      limits = c(0, 100)) +
ggplot2::geom_sf(data = uruguay, fill = NA, color = "black", inherit.aes = FALSE)

```



1278

```

#cowplot::ggdraw() +
#  cowplot::draw_plot(tmax_plot, x = 0, y = 0.5, width = 1, height = .5) +
#  cowplot::draw_plot(tmin_plot, x = 0, y = 0, width = 1, height = .5)

```

1279 En el panel superior se muestra la temperatura máxima observada (negra) y las temperaturas

₁₂₈₀ sintéticas (naranja) para el año 2019. En el inferior se muestra la temperatura mínima diaria
₁₂₈₁ observada (negra) y cada una de las realizaciones (azul).

₁₂₈₂ **References**

- ₁₂₈₃ Kleiber, W., Katz, R.W., Rajagopalan, B., 2012. Daily spatiotemporal precipitation simu-
₁₂₈₄ lation using latent and transformed gaussian processes. Water Resources Research 48.
- ₁₂₈₅ Kleiber, W., Katz, R.W., Rajagopalan, B., 2013. Daily minimum and maximum temperature
₁₂₈₆ simulation over complex terrain. Ann. Appl. Stat. 7, 588–612.
- ₁₂₈₇ Simpson, G., 2018. Introducing *gratia*. From the Bottom of the Heap.
- ₁₂₈₈ Verdin, A., 2016. Stochastic space-time modeling for agricultural decision support in the
₁₂₈₉ argentine pampas (Thesis).
- ₁₂₉₀ Wickham, H., 2011. *Ggplot2*. Wiley Interdisciplinary Reviews: Computational Statistics 3,
₁₂₉₁ 180–185.
- ₁₂₉₂ Wood, S.N., 2001. *Mgcv*: GAMs and generalized ridge regression for r. R news 1, 20–25.
- ₁₂₉₃ Wood, S.N., 2017. Generalized additive models: An introduction with r. Chapman;
₁₂₉₄ Hall/CRC.